GRÁFICAS Y FUNCIONES III

Al igual que las dos últimas prácticas, se realizó un ejercicio en el que se aplicaron comandos que ya aprendimos (funciones, gráficas, bucles...). En este recurso se explica cómo realizar el ejercicio paso a paso.

ÍNDICE

ERRORES A EVITAR

1. **En funciones, recordar llamar a la función.**

   Cuando trabajamos con funciones debemos llamar a la función, ya que si no lo hacemos, no ocurre nada.

2. **Para los vectores ss y ff, debemos usar runif**

   Al pedirnos que le asignemos valores aleatorios, no podemos usar el comando seq, ya que este nos da valores equidistantes entre sí y no al azar.

EJERCICIO

Dividiremos el ejercicio en dos partes (relacionadas entre sí).

PRIMERA PARTE:

Se considera una sustancia cuya densidad viene dada por d(x) siendo x la coordenada espacial. La masa total en cierto intervalo [A, B] está dada por:

$$M_{AB} = \int_A^B d(x)dx$$

Se desea realizar un programa en R para resolver dicha integral mediante una fórmula de Simpson compuesta, cuya expresión viene dada por:

$$\int_A^B d(x)dx \approx \frac{h}{6}\left(d(A) + 2\sum_{i=2}^{n-1} d(T_i) + 4\sum_{i=1}^{n-1} d\left(\frac{T_i + T_{i+1}}{2}\right) + d(B)\right) *$$

Donde se ha realizado una subdivisión del intervalo [A, B] en (n-1) subintervalos iguales, es decir, considerando n puntos Ti, y siendo h la longitud de cada subintervalo.

El programa se ejecutará con los siguientes datos: A=0, B=3.05, densidad dada por d(x)=exp(x)sin(x). Como número de puntos para el desarrollo de la fórmula se tomará n=35.

NOTAS:

1. Se programará la expresión de la densidad en una función llamada d.
2. La fórmula de Simpson compuesta se programará en otra función llamada Simpson.
3. Los sumatorios se realizarán con bucles condicionales.
4. El resultado exacto de la masa es: M= 11.97907159…

Por último, haremos una representación gráfica de la función de densidad obtenida por el método de Simpson.

VARIABLES YA ASIGNADAS

- A=0
- B=3.05
- n=35

PASOS A SEGUIR:

1. Definir el vector T, donde se almacenan n puntos del intervalo [A, B] equidistantes entre sí.
2. Definir la longitud de los subintervalos como h. (Intervalo [A, B] dividido en (n-1) subintervalos)
3. Definir la función d(x)=exp(x)sin(x)
4. Programar la función de Simpson dada por la expresión de la fórmula de Simpson (*)
5. Almacenar en Ms el valor de la función Simpson para n, T, h, A y B
6. Representar la gráfica de la función obtenida por el método de Simpson

```
par(mfrow=c(2,1))
x=seq(A,B,0.01)
plot(x,d(x[1:length(x)]),xlim=c(A,B),ylim=c(0,8),xlab='x',ylab='rho')
par(new='TRUE')
plot(T,d(T[1:length(T)]),xlim=c(A,B),ylim=c(0,8),type='h',col='green',xlab='',ylab='')
```

SOLUCIÓN PRIMERA PARTE:

```
A=0
B=3.05
n=35
T=seq(A,B,length=n)
h=(B-A)/(n-1)
d=function(x){exp(x)*sin(x)}
```

*Introducimos y/o asignamos las variables (**A,B,n,h**), vector (**T**) y función (**d**) que usaremos*

```
Simpson=function(n,T,h,A,B){
        d1=0
        d2=0
```

*La función Simpson va a depender de 5 valores (**n, T, h, A, B**).*
*Inicializamos **d1 y d2** a 0 (d1 y d2 se usarán para realizar el sumatorio)*

```
        for(i in 2:(n-1)){
                d1=d(T[i])+d1
        }
```

*Abrimos un **bucle** que va desde 2 hasta (n-1) (siendo n el nº de puntos que se ha tomado para realizar la fórmula)*

*En **d1** se sumará cada valor de densidad que tiene cada una de las componentes de T, es decir, es un **sumatorio** de d(T[i]).*

```
        for(i in 1:(n-1)){
                d2=d((T[i]+T[i+1])/2)+d2
        }
```

*Abrimos otro **bucle** que va desde 1 hasta (n-1) (siendo n el nº de puntos que se ha tomado para realizar la fórmula)*

```
        M=h/6*(d(A)+2*d1+4*d2+d(B))
```

*Asignamos a una variable **M** la función de Simpson que es **h/6*(d(A)+2*d1+4*d2+d(B))***

```
        return(M)
```

*Usamos el comando **return()** para que nos de los valores de **M***

#Vamos a llamar **Ms** a la variable que tiene el valor para **n=35, T=seq(A, B, length=n), h=(B-A)/(n-1), A=0, B=3.05**, como ya hemos introducido estos valores al principio, con escribir "**n, T, h, A, B**" nos vale, ya que tiene este valor asignado.

```
}

Ms=Simpson(n,T,h,A,B)

par(mfrow=c(2,1))
```

*Usamos este comando para decir que salgan **dos gráficos a la vez**, uno encima de otro*

```
x=seq(A,B,0.01)
```

*Utilizamos el comando **seq** para obtener un vector cuyo primer componente sea **A**, el último **B** y el incremento **0.01***

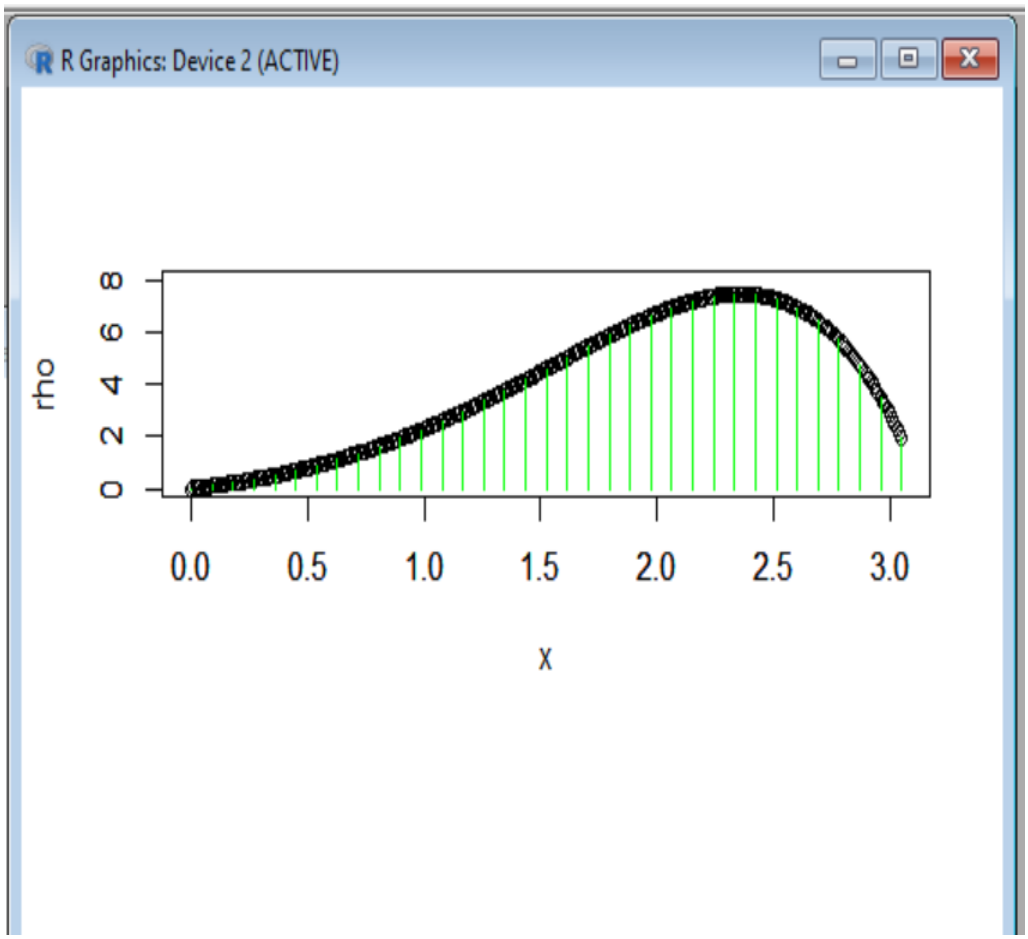
```
plot(x,d(x[1:length(x)]),xlim=c(A,B),ylim=c(0,8),xlab='x',ylab='rho')

par(new='TRUE')
```

*Para representar los valores de densidad de los componentes del vector **T** en función de las abscisas x.*

plot(T,d(T[1:length(T)]),xlim=c(A,B),ylim=c(0,8),type='h',col='green',xlab='',ylab='')

*Ponemos en ambos **xlim=c(A,B), ylim= c(0,8)** para que la escala sea la misma*

Obtendremos una gráfica tal que así:



SEGUNDA PARTE:

En esta parte, haremos la gráfica de la función otra vez pero por el método de Montecarlo y también representaremos la de Simpson para compararlas.

PASOS A SEGUIR:

1. Introducir una variable denominada num_puntos que contiene el número de valores a considerar (p.ej. 1000 puntos y luego incrementar).
2. Definir extremos del dominio de cálculo: AA=0; BB=7.4 ; A=0; B=3 (en nuestro caso).
3. Generar los vectores ss, ff que contengan num_puntos aleatorios en el intervalo [A,B] y [AA,BB] respectivamente.
4. Inicializar los vectores puntos_dentro=0; puntos_fuera=0; color=0
5. Para i desde 1 hasta num_puntos

   Si (ff[i]<=d(ss[i]) entonces

   puntos_dentro= puntos_dentro+1

   color[i]='blue'

   si no

   puntos_fuera= puntos_fuera+1

   color[i]='orange'

   Fin condición

   Fin bucle
6. Representar conjuntamente ambas zonas (dentro y fuera del área)

   plot(ss,ff,xlim=c(A,B),ylim=c(AA,BB),col=color,ylab='Densidad',xlab='x')

   par(new='TRUE')

   ss=seq(A,B,0.001)

   plot(ss,d(ss[1:length(ss)]),type='l',col='dark green', xlim=c(A,B), ylim=c(AA,BB), xlab='', ylab='')
7. Aproximar el valor de la integral: Area=puntos_dentro/num_puntos*B*BB
8. Introducir el valor exacto de la integral: Vexact=11.97907159
9. Obtener los errores cometidos con la fórmula de Simpson Compuesta y el Método de Montecarlo

   ErrorM=(Area-Vexact)/Vexact

   ErrorS=(Masa-Vexact)/Vexact
10. Escribir resultados en forma de tabla (estructura data.frame)

    ErrorM=(Area-Vexact)/Vexact; ErrorS=(Masa-Vexact)/Vexact

    Método=c("Valor exacto","Simpson","Montecarlo")

    Valores=c(Vexact,Masa,Area)

    Error_Relativo=c(0,ErrorS,ErrorM)

    data.frame(Método,Valores,Error_Relativo)

SOLUCIÓN SEGUNDA PARTE:

```
num_puntos=1000
AA=0
BB=7.4
A=0
B=3
```

*Introducimos la variable **num_puntos** y definimos los límites del dominio de cálculo (**AA=0, BB=7.4**) y (**A=0,B=3**)*

```
ss=runif(num_puntos,A,B)
ff=runif(num_puntos,AA,BB)
```

*Generamos el vector **ss** que contiene num_puntos **aleatorios** en el intervalo **[AA,BB]**, y el vector **ff** que contiene num_puntos **aleatorios** en el intervalo **[A,B]***

```
puntos_dentro=0
puntos_fuera=0
color=0
```

*Inicializamos a 0 los vectores que van a entrar en el bucle (**puntos_dentro, puntos_fuera, color**). Si no lo hacemos, el programa no hallará el vector y dará **error***

```
for(i in 1:num_puntos){
```

Abrimos un bucle que irá desde 1 hasta num_puntos (nº de valores a considerar)

```
        if(ff[i]<=d(ss[i])){
```

SEGUIMOS DENTRO DEL BUCLE
*Abrimos una estructura condicional en el que si se cumple que **ff[i]** es menor o igual que **d(ss[i])** entonces sumaremos 1 a **puntos_dentro** y haremos que la componente i del vector **color** sea **'blue'**.*

```
                puntos_dentro=puntos_dentro+1
                color[i]='blue'
```

*Si no se cumple la condición, sumaremos 1 a **puntos_fuera** y haremos que la componente i del vector **color** sea **'orange'**.*
Cerramos condición
Cerramos bucle

```
        }else{
                puntos_fuera=puntos_fuera+1
```

```
            color[i]='orange'
        }
}
```

*Para representar los valores de densidad de los componentes del vector **ss** en función de las abscisas ss.*

```
plot(ss,ff,xlim=c(A,B),ylim=c(AA,BB),col=color,xlab='x',ylab='Densidad')
```

```
par(new='TRUE')ss=seq(A,B,0.001)
plot(ss,d(ss[1:length(ss)]),type='l',col='dark
```

*Ponemos en ambos **xlim=c(A,B), ylim= c(AA,BB)** para que la escala sea la misma*

```
green',xlim=c(A,B),ylim=c(AA,BB),xlab='',ylab='')
```

```
Area=puntos_dentro/num_puntos*B*BB
```

*Aproximamos el valor de la integral dividiendo **puntos_dentro** entre **num_puntos** por **B** y por **BB**. Luego, la guardamos en **Area***

```
Vexact=11.97907159
```

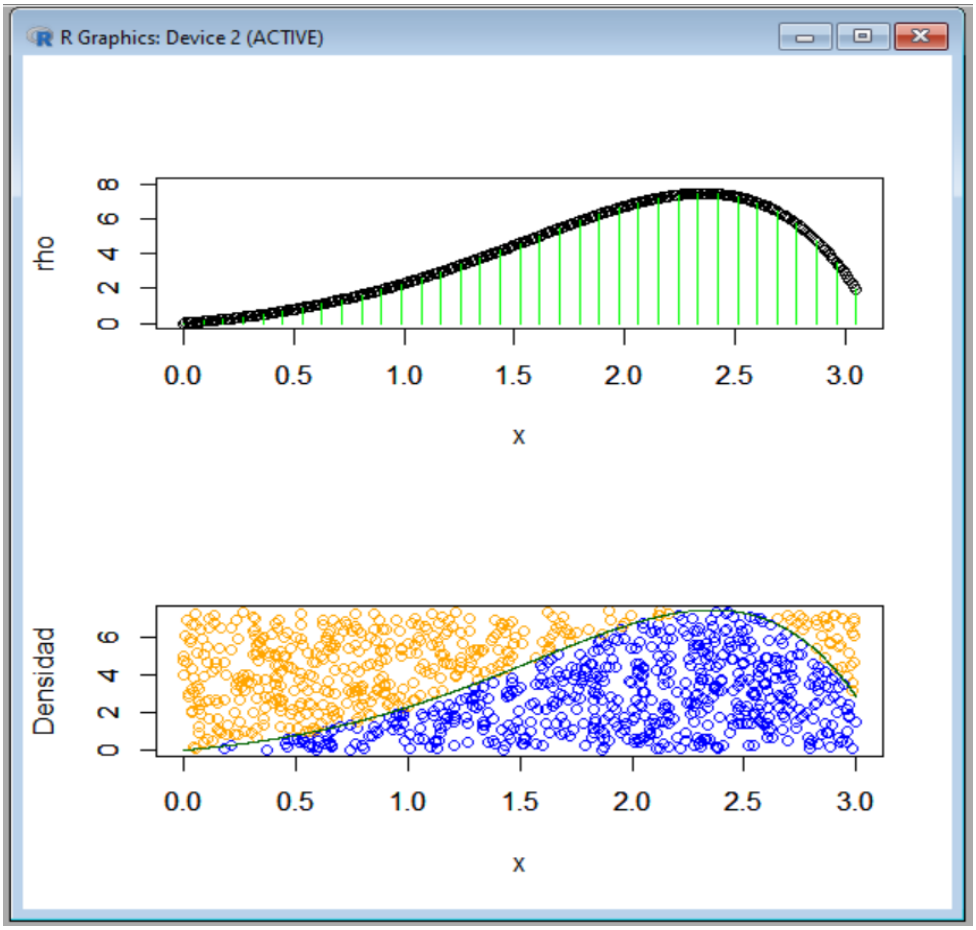
Introducimos el valor exacto de la integral que es **11.97907159**

```
ErrorM=(Area-Vexact)/Vexact
ErrorS=(Ms-Vexact)/Vexact
```

Calculamos los errores cometidos en la fórmula de Montecarlo y la de Simpson

```
Metodo=c('Valor
exacto','Simpson','Montecarlo')
Valores=c(Vexact,Ms,Area)
Error_Relativo=c(0,ErrorS,ErrorM)
data.frame(Metodo,Valores,Error_Relativo)
```

Por último, construimos una tabla en la que pondremos el método por el que se ha obtenido, los valores que han dado y los errores que se ha cometido



	Metodo	Valores	Error_Relativo
1	Valor exacto	11.97907	0.000000e+00
2	Simpson	11.97907	-9.019708e-08
3	Montecarlo	12.36540	3.225028e-02

EJERCICIO COMPLETO

```
#Ejercicio 2
A=0
B=3.05
n=35
T=seq(A,B,length=n)
h=(B-A)/(n-1)

 d=function(x){exp(x)*sin(x)}
Simpson=function(n,T,h,A,B){
        d1=0
        d2=0
        for(i in 2:(n-1)){
                d1=d(T[i])+d1
        }
        for(i in 1:(n-1)){
                d2=d((T[i]+T[i+1])/2)+d2
        }
        M=h/6*(d(A)+2*d1+4*d2+d(B))
        return(M)
}
Ms=Simpson(n,T,h,A,B)

par(mfrow=c(2,1))
x=seq(A,B,0.01)
plot(x,d(x[1:length(x)]),xlim=c(A,B),ylim=c(0,8),xlab='x',ylab='rho')
par(new='TRUE')
plot(T,d(T[1:length(T)]),xlim=c(A,B),ylim=c(0,8),type='h',col='green',,xlab='',ylab='')

num_puntos=1000
AA=0
BB=7.4
A=0
B=3
ss=runif(num_puntos,A,B)
ff=runif(num_puntos,AA,BB)
puntos_dentro=0
puntos_fuera=0
color=0
```

```
for(i in 1:num_puntos){
        if(ff[i]<=d(ss[i])){
                puntos_dentro=puntos_dentro+1
                color[i]='blue'
        }else{
                puntos_fuera=puntos_fuera+1
                color[i]='orange'
        }
}

plot(ss,ff,xlim=c(A,B),ylim=c(AA,BB),col=color,xlab='x',ylab='Densidad')
par(new='TRUE')
ss=seq(A,B,0.001)
plot(ss,d(ss[1:length(ss)]),type='l',col='dark green',xlim=c(A,B),ylim=c(AA,BB),xlab='',ylab='')

Area=puntos_dentro/num_puntos*B*BB
Vexact=11.97907159
ErrorM=(Area-Vexact)/Vexact
ErrorS=(Ms-Vexact)/Vexact

Metodo=c('Valor exacto','Simpson','Montecarlo')
Valores=c(Vexact,Ms,Area)
Error_Relativo=c(0,ErrorS,ErrorM)
data.frame(Metodo,Valores,Error_Relativo)
```