GRÁFICAS Y FUNCIONES II

Esta práctica, al igual que la anterior, consistió en un ejercicio en el que se aplicaron conceptos vistos anteriormente (funciones, gráficas, bucles…). En este recurso se explica como realizar el ejercicio paso a paso.

ÍNDICE

TEORÍA

Nuevos comandos

• **xlim=c(),ylim=c()** → definir los límites.

Lo usamos dentro del comando plot.

Sumatorios

Los **sumatorios (Σ)** se tratan de un tipo de operación que consiste en la suma de todos los elementos de una serie de datos. Podremos hacer esta operación tanto con vectores como con matrices.

• **Inicializar a 0** (si a un número le sumamos 0, obtenemos el mismo número)

Productorios:

Los **productorios (Π)** son operaciones muy similares a los sumatorios, con la diferencia de que, en lugar de sumar los elementos, los multiplican.

• **Inicializar a 1** (si un número lo multiplicamos por 1, obtenemos le mismo número)

CONSEJOS Y 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. **En funciones, asignar valores en el orden adecuado.**

Es muy importante que después, al llamar a la función tras darle un valor a la variable, introduzcamos las variables en el mismo orden que habíamos puesto en la función:

```
Polbase=function(s,t,n){
……
}
```
Cuando después demos un valor a t y almacenemos ese valor en A1 tenemos que poner:
```
A1=Polbase(s,t,n)
```

(En este caso el orden es s, después t y por último n; tenemos que respetar ese orden)

3. **En funciones, introducir tantos valores como valores de entrada tiene la función:**

Si la función tiene por ejemplo tres variables, debemos poner la tres:

Polbase(s,2) MAL

Polbase(s,2,n) BIEN

4. A la hora de representar en la gráfica:

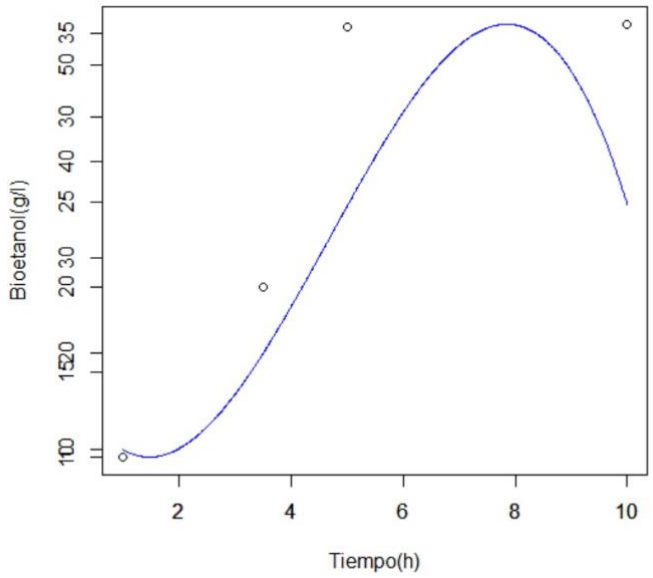
Dentro del comando plot, usamos xlim y ylim para delimitar la gráfica:

- **Sin xlim y ylim**

```
plot(s,B,xlab='Tiempo(h)',ylab='Bioetanol(g/l)')  par(new='TRUE')

plot(x,f,xlab='',ylab='',col='blue',type='l')
```

La gráfica nos queda:

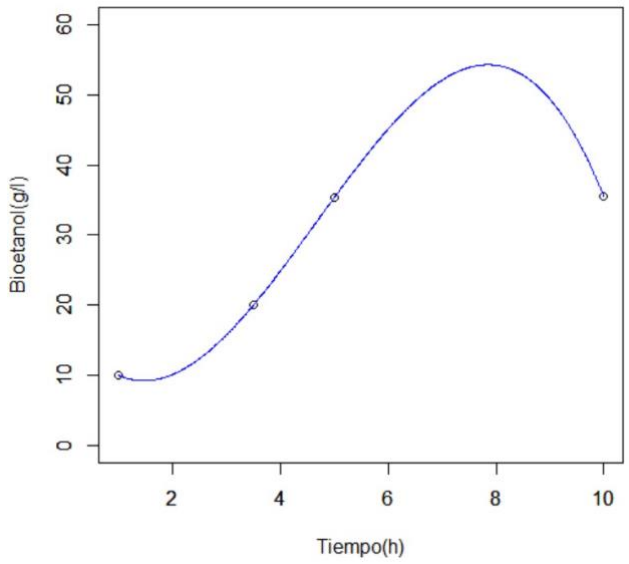


- **Con xlim y ylim**

plot(s,B,xlim=c(s[1],s[n]),ylim=c(0,60),xlab='Tiempo(h)',ylab='Bioetanol(g/l)') par(new='TRUE')

plot(x,f,xlim=c(s[1],s[n]),ylim=c(0,60),xlab='',ylab='',col='blue',type='l')

La gráfica nos queda:



EJERCICIO

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

PRIMERA PARTE:

Se conoce la producción de bioetanol que se obtiene en una planta de biocombustibles en función del tiempo. Los valores de la concentración (g/l) estan almacenados en el vector: B(10,20,35.33,35.5) y los instantes de tiempo (horas) en el vector s(1,3.5,5,10).

SE PIDE: Realizar un script llamado Bioetanol.R para estimar, mediante interpolación de Lagrange, la concentración de bioetanol en los instantes t = 2 y t = 6.5. Para ello, se empleará la siguiente expresión del polinomio interpolador de Lagrange.

$$p = \sum_{i=1}^{n} B_i L_i$$

PASOS:

1. Programar la función Polbase para obtener las funciones de base la interpolación de Lagrange, que están dadas por la expresión:

$$L_i = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{t - s_j}{s_i - s_j}, (i = 1, \ldots, n)$$

   L es un vector cuyas componentes son los valores de cada polionomio de base en el punto t.

2. Programar la función PolInterp para obtener el polinomio interpolador en el punto t. Dicha función recibirá como argumentos de entrada:
   • un vector B (que contiene los valores de la función que se interpola).
   • el vector L (que contiene los polinomios de base de Lagrange evaluados en el punto t).
   • la variable n (número de puntos del soporte de interpolación).

3. Introducir los datos y ejecutar las funciones programadas. Al ejecutar, se almacenarán las funciones de base en el vector L y el polinomio interpolador en la variable p.

SOLUCIÓN PRIMERA PARTE:

```
s<-c(1,3.5,5,10)
n<-length(s)
B<-c(10,20,35.33,35.5)
t<-2
```

*Introducimos los datos que vamos a necesitar. **B** y **s** son **vectores**, por lo que los escribimos con su estructura correspondiente.*

***n** representa la longitud del vector s.*

```
#FUNCIÓN POLBASE
Polbase<-function(s,t,n){
L=0
```

La función Polbase va a depender de 3 valores (s,t,n).

***IMP**: Inicializamos el vector L a 0*

```
for (i in 1:n){
  L[i]=1
```

*Abrimos un **bucle** que irá desde **1** hasta **n** (número de componentes de s).*

*Se asignará el valor 1 a cada componente de nuestro vector L para poder realizar el **productorio** más tarde.*

```
    for (j in 1:n){
      if(j!=i){
  L[i]=L[i]*((t-s[j])/(s[i]-s[j]))
        }
    }
 }
return(L)
}
```

SEGUIMOS DENTRO DEL BUCLE 1

*Abrimos un **segundo bucle** que irá también de **1** a **n**.*

*En caso de que **se cumpla la condición** (j debe ser diferente de i) se **realiza la operación.** De esta manera se irán obteniendo los valores del vector L.*

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

#FUNCIÓNPOLINTERP

```
PolInterp<-function(B,L,n){
 p<-0


  for(i in 1:n){
     p<-p+B[i]*L[i]
      }
return(p)
}
```

Repetimos en mismo proceso, esta vez para t=6.5

La función PolInterp va a depender de 3 valores (B,L,n).

***IMP**: Inicializamos el polinomio p a 0*

*Abrimos un **bucle** que irá de **1** hasta **n**.*

*Se realizará la **operación** para obtener los **componentes del polinomio**.*

*Usamos el comando **return** para que nos de nuestro polinomio.*

#Vamos a llamar **L2 al polinomio de base para t=2**, como ya hemos introducido que t vale 2 al principio, con escribir "t" nos vale, ya que tiene este valor asignado; más tarde, cuando quiera calcularlo en t=6.5 deberé cambiar el valor de t.

```
L2<-Polbase(s,t,n)
L2


Inter2<-PolInterp(B,L2,n)
Inter2
```

Una vez calculado L2 calculamos el polinomio interpolador en t=2 al que llamamos Inter2.

Teníamos que PolInterp dependía de B, L y n. Mantenemos B y n, pero en L tenemos que introducir L2 (que acabamos de calcularlo)

#Vamos a **llamar L6.5 al polinomio de base para t=6.5**, como el valor que tiene t ahora mismo es 2, tengo que especificar que quiero el polinomio en el que t vale 6.5.

```
L6.5<-Polbase(s,6.5,n)
L6.5
```

```
Inter6.5<-PolInterp(B,L6.5,n)
Inter6.5
```

Para representar los valores de B en función de las abscisas s.

Usamos par=(new="TRUE") para superponer curvas.

SEGUNDA PARTE:

Queremos ahora dibujar conjuntamente los puntos y el polinomio interpolador

4. Obtener 1001 abscisas equidistantes en el intervalo [s[1],s[n]] y almacenarlas en un vector x. (Calcularemos el valor del polinomio interpolador en cada abscisa para dibujar).

5. Llamamos 1001 veces a las funciones Polbase y PolInterp. La salida de Polinterp se guardará en un vector f[k], k=1,…,1001.

SOLUCIÓN SEGUNDA PARTE

```
x=seq(s[1],s[n],length=1001)
```

Utilizamos el comando seq para obtener un vector cuyo primer componente sea s[1], el último s[n] y la longitud 1001 (ya que ese es el número de componentes que nos piden).

```
f=0
for (k in 1:1001){
 A<-Polbase(s,x[k],n)
  f[k]<-PolInterp(B,A,n)
 }
f
```

Guardamos en A el polinomio de base y vamos a calcular con este el polinomio interpolador.

plot(**s,B**,xlim=c(s[1],s[n]),ylim=c(0,60))

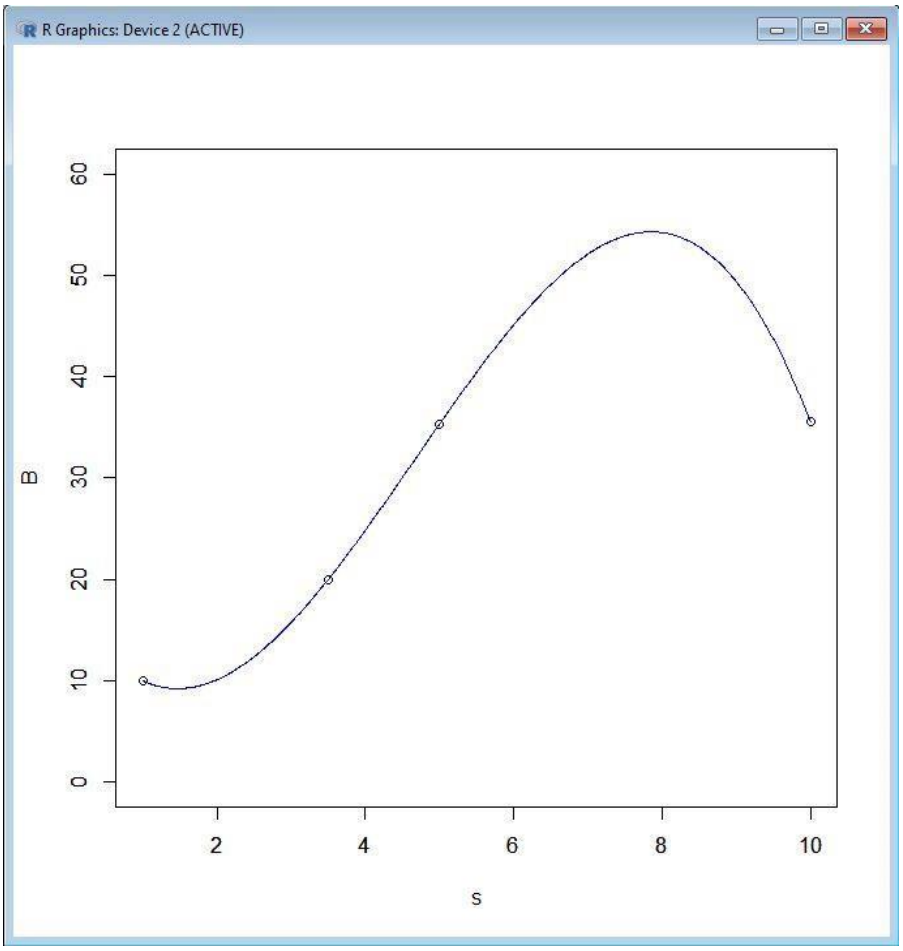
par=(new="TRUE")

Para representar los valores de f en función de las abscisas x.

plot(**x,f**,xlim=c(s[1],s[n]),ylim=c(0,60),type="l",col="blue", xlab="",ylab="")

Ponemos en ambos ylim= c(0,60) para que la escala sea la misma

Vamos a obtener la **gráfica**:



EJERCICIO COMPLETO

```
s<-c(1,3.5,5,10)
n<-length(s)
B<-c(10,20,35.33,35.5)
t=2

Polbase<-function(s,t,n){
L=0
for (i in 1:n){
  L[i]=1
   for (j in 1:n){
      if(j!=i){
     L[i]=L[i]*((t-s[j])/(s[i]-s[j]))
    }
   }
 }
return(L)
}

PolInterp<-function(B,L,n){
  p<-0
   for(i in 1:n){
     p<-p+B[i]*L[i]
   }
 return(p)
}

L2<-Polbase(s,t,n)
L2
```

```
Inter2<-PolInterp(B,L2,n)
Inter2


L6.5<-Polbase(s,6.5,n)
L6.5

Inter6.5<-PolInterp(B,L6.5,n)
Inter6.5


x=seq(s[1],s[n],length=1001)

f=0
for (k in 1:1001){
  A<-Polbase(s,x[k],n)
   f[k]<-PolInterp(B,A,n)
 }
f

plot(s,B,xlim=c(s[1],s[n]),ylim=c(0,60))
par=(new="true")
plot(x,f,xlim=c(s[1],s[n]),ylim=c(0,60),type="l",col="blue", xlab="",ylab="")
```