Biotecnología Programación 2023-2024 Dificultad: media-alta
1º curso Grupo M10

# PRÁCTICA 5

En este documento hay una guía para entender cómo resolver el ejercicio de la práctica 5:

## INTERPOLACIÓN

**Enunciado**:

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) están 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.

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

En primer lugar interpretaremos los que nos están pidiendo; nos proporcionan los valores de una función que no conocemos, en cuatro puntos diferentes. Conocemos que en el vector ***s*** se almacenan las abscisas, y en el vector ***B*** las ordenadas. Nos piden estimar los valores de dicha función para t=2 y t=6.5. Nos piden hacerlo de dos formas diferentes, en primer lugar utilizando funciones de base, y en segundo lugar utilizando la fórmula de Newton en cada subintervalo.

## RESOLUCIÓN MEDIANTE FUNCIONES DE BASE

### PASO 1: Crear la función Polbase()

Vamos a crear una función que partiendo de los datos que nos facilita el enunciado, nos proporcionará los polinomios de base Lagrange ($L_i$ en el enunciado). Para ello crearemos una función llamada Polbase, que nos devolverá las funciones de base de la interpolación de Lagrange dadas por la expresión:

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

Siendo L un vector cuyas componentes son los valores de cada polinomio de base en el punto t. Observamos que variables tenemos en la fórmula (a la derecha del signo igual), vemos que son ***n, t*** y el vector ***s*** dependiente de ***i*** y ***j.*** Utilizaremos por tanto dos bucles, como se impone la condición de que j≠ i, tendremos que aplicar un condicional que defina la función. Definimos $L_i$ y utilizamos return(L) para que el programa nos devuelva los resultados.

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

**PASO 2**: **Crear función PolInterp()**

Crearemos una segunda función a la que llamaremos PolInterp, con el polinomio que nos proporciona el enunciado:

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

Esta función tendrá como argumentos; el vector ***B***, que tiene los valores de la función que se va a interpolar, el vector ***L***, que tienen los polinomios de base, y la variable ***n***, es decir el nº de puntos del soporte de interpolación.
PolInterp calculará el valor aproximado de la función al darle el punto, y las ordenadas de las abscisas del apartado anterior. En este caso no será necesario tener en cuenta el orden en el que se proporcionan los datos.

```
polinterp = function(B,L,n){
        p = 0
        for (i in 1:n){
                p = p + B[i]*L[i]
        }
return(p)
}
```

**PASO 3**: **INTRODUCCIÓN DATOS**

A continuación introduciremos los datos proporcionados en el enunciado.
Ejecutaremos las funciones para los valores solicitados (2 y 6.5), y obtendríamos el resultado.

Como debemos calcular Polbase para cada valor solicitado y a continuación con dicho resultado calcular PolInterp también para cada valor, podemos automatizar el proceso, aunque en este caso al ser solo dos valores lo hemos hecho manualmente.

```
# DATOS
B=c(10, 20, 35.33, 35.5) ; s=c(1, 3.5, 5, 10) ; n=length(B) ; t=c(2, 6.5)
#        Ordenadas              Abscisas     Para el grado    Nos piden

Lbase1=Polbase(s, t[1], n)   # t=2
Lbase2=Polbase(s, t[2], n)   # t=6.5

Pol1=PolInterp(B, Lbase1, n) # t=2
Pol2=PolInterp(B, Lbase2, n) # t=6.5

Pol1
Pol2
```

Una vez ejecutadas las funciones, debemos almacenar el valor solución en una dirección de memoria, para lo que elegiremos un nombre para cada función (Lbase1=Polbase (s,t(1),n), Lbase2=Polbase(s,t(2),n), Pol1=PolInterp(B,Lbase1,n), Pol2=PolInterp(B,Lbase2,n)).

**RESOLUCIÓN MEDIANTE FÓRMULA DE NEWTON**

**PASO 4**: **Crear la función Polinomio**

En este caso vamos a utilizar la fórmula de Newton para calcular el polinomio interpolador. Para ello necesitaremos calcular las diferencias divididas, para lo cual utilizaremos la siguiente fórmula:

$$P = f\left[s_0\right] + f[s_0, s_1] * [t - s_0]$$

Donde:

$B_0$= f[$s_0$]=sumando1
$B_1$-$B_0$/$s_1$-$s_0$= f[$s_0$, $s_1$]=factor 1
(t-$t_0$)=factor 2

Usaremos esta fórmula ya que no podemos interpolar todos los puntos juntos, sino que debemos proceder tramo a tramo.
Creamos la función Polinomio, que contendrá todos los datos que conocemos, y las abscisas del punto que queremos estimar. Introduciremos un bucle *while* para saber en qué tramo nos encontramos. Primero estableceremos que la variable trozo es igual a uno, así comenzamos en el primer intervalo, e impondremos la condición de que el programa vaya comprobando si el valor de x

supera el extremo del intervalo. Si no lo supera se encuentra dentro del intervalo, y en caso de que lo supere, debemos pasar al siguiente intervalo, sumando 1 a la variable trozo.

Una vez conocido en que tramo nos encontramos, aplicamos la fórmula. Usaremos el comando *return,* para obtener el valor estimado en este punto aplicando la fórmula de Newton.

```
Polinomio = function(x, s, B) {
  trozo=1
  while (x>s[trozo+1]) {trozo=trozo+1}
  sumando1=B[trozo]
  factor1=(B[trozo+1]-B[trozo])/(s[trozo+1]-s[trozo])
  factor2=x-s[trozo]
  return(sumando1+factor1*factor2)
}
```

A continuación ejecutaremos la función Polinomio pata t=2 y t=6.5.

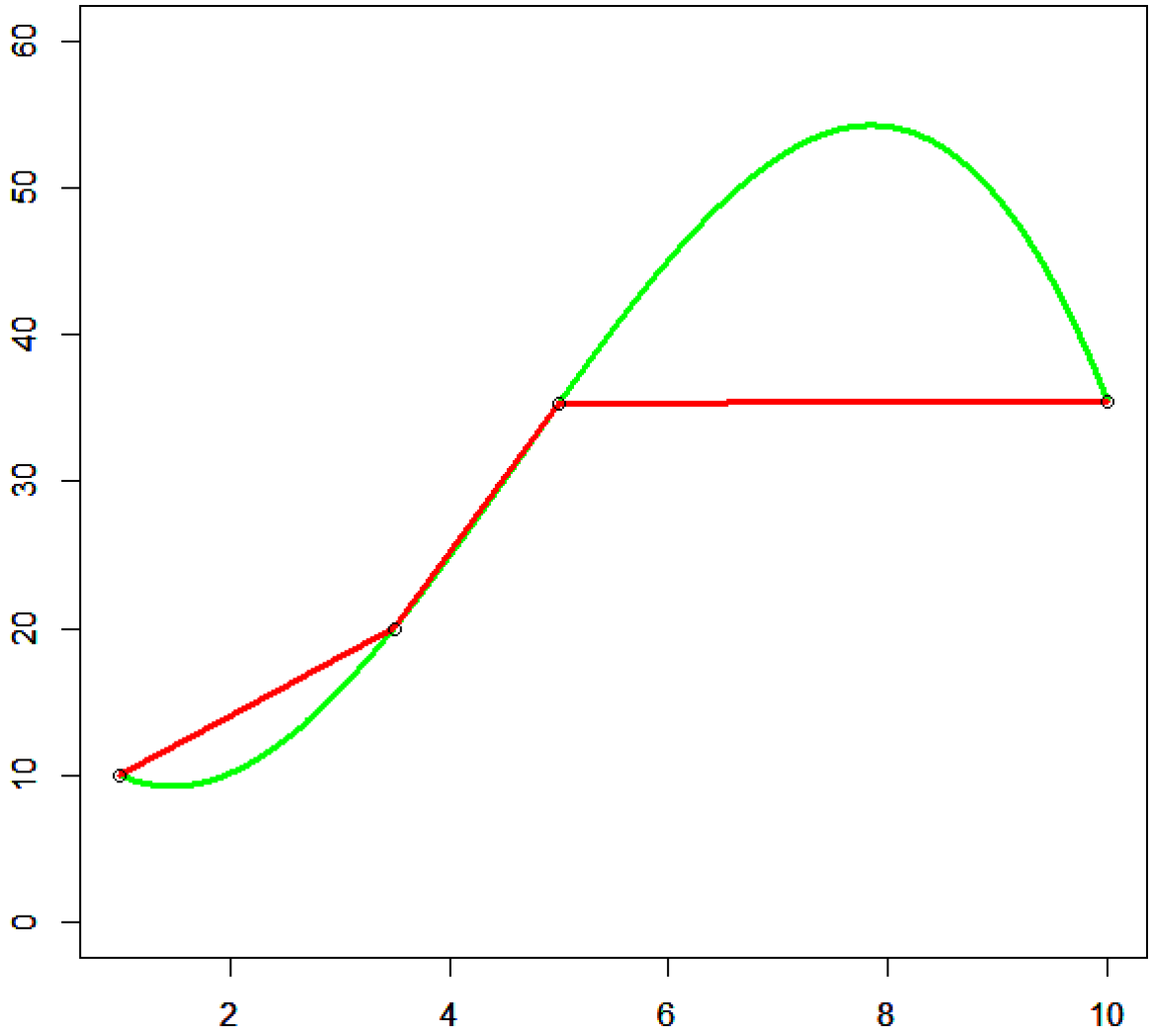
```
print(Polinomio(2, s, B))
print(Polinomio(6.5, s, B))
```

**PASO 5**: **Crear la gráfica con un *plot()***

Para analizar los datos obtenidos y sacar conclusiones, representamos los valores obtenidos por ambos métodos en una misma gráfica. Obtendremos 1001 puntos dentro del intervalo que conocemos de cada función. Usaremos *seq* para obtener los valores de abscisas. Mediante un bucle *for* obtendremos el valor de cada función en los diferentes puntos. Las imágenes obtenidas mediante el método 1 las guardaremos en un vector *f*, y las obtenidas en el método 2 en un vector *g*.

```
x=seq(s[1], s[n], length=1001)  # x son las abscisas de los valores de f y g
f=0                             # f son las imágenes de x por el método 1
g=0                             # g son las imágenes de x por el método 2
for (k in 1:1001) {
  apoyo=Polbase(s, x[k], n)          # Paso 1 del método 1
  f[k]=PolInterp(B, apoyo, n)        # Paso 2 del método 1
  g[k]=Polinomio(x[k], s, B)         # Método 2 (se hace todo en un paso)
}
```

Utilizaremos  el comando *plot* para representar gráficamente las funciones, introduciremos en primer lugar las abscisas, a continuación las ordenadas, y después el resto de los datos.
Como vamos a superponer ambas funciones, debemos establecer un límite vertical mediante el comando *ylim*.
Para que las gráficas se superpongan utilizaremos *par (new="true").*




La función verde corresponde al método 1 y la roja al método 2. Ambas funciones coinciden en los puntos de apoyo, pero son muy diferentes. La función verde en algunos tramos se aleja mucho de los valores dados, por lo que pensamos que la roja es más fiable, aunque tenga una forma poco común. No podemos determinar cuál de las dos es correcta ya que no conocemos la función original.