Cómo hacer una interpolación de Lagrange en R.

En esta entrada veremos paso a paso cómo hacer una interpolación según el método de Lagrange.
La interpolación consiste en, a partir de una serie de datos conocida, encontrar una función/polinomio que nos permita estimar otros valores desconocidos de esa misma serie de datos. Se trata de una herramienta muy útil a la hora de trabajar con R, y puede aparecer en diversos ejercicios y exámenes.

Para este tipo de ejercicios es necesario tener conocimientos previo sobre funciones, bucles for, sumatorios y productorios.

La fórmula general del polinomio de interpolación de Lagrange es:

$$p(x) = \sum_{i=1}^{n} B_i L(x)$$

A la hora de resolver estos ejercicios, contaremos con una serie de datos iniciales:

- Una **serie de puntos** que forman la variable independiente **x** (por ejemplo, el tiempo en el que se han medido unas muestras). Reciben el nombre de puntos de soporte, ya que a partir de ellos calcularemos la función de los polinomios de base.
- Una **serie de puntos** que forman la variable dependiente de x **y** (por ejemplo, la temperatura que había en un tiempo determinado, o la población de individuos que hay en un momento determinado)
- **Valores de x a interpolar**, es decir, cuya y es desconocida y que calcularemos a partir de la función. **(t)**

Según la notación de la fórmula anterior, llamaremos B al vector con las componentes y **(B=y)**. Así mismo, vamos a renombrar al vector que contiene los valores de X como S **(x=s)** Además, n será el número de puntos que tengamos, y condicionarán el grado del polinomio resultante (siendo este n-1).

Por tanto, lo que necesitamos calcular son nuestros valores de L. L representa la función de polinomios de base, cuya expresión es:

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

Como podéis ver, tenemos que calcular 2 funciones, por lo que nuestro ejercicio consistirá en calcular estas dos funciones, para finalmente representarlas con diversos valores de t.

Pasamos ahora a ver los pasos para resolver el ejercicio:

- Primero debemos definir la función Polbase, que contendrá los valores de la función L. Esta función, como se ve en la fórmula general, depende de las variables n,t,s, por lo que esas serán las que tenemos que introducir en el comando function.
- Dentro de la función abriremos un bucle for para el subíndice [i], variando desde 1 hasta n. En este punto conviene también inicializar nuestro vector L a 1.
- Dentro del bucle, abrimos otro bucle for para el subíndice [j], variando también de 1 hasta n. Sin embargo, como se ve en la fórmula, la i y la j no pueden ser iguales ($i \neq j$) por lo que antes de continuar tendremos que abrir un bucle if en el que especificaremos que la operación sólo se haga si i es distinto de j **(i != j).**
- Dentro de esta condición, indicaremos la operación del productorio para obtener L, según la fórmula indicada anteriormente.
- Cerramos ahora todos nuestros bucles, y antes de cerrar la función indicamos "return(L)", esto es importante para que la función realmente obtenga los valores de L.

Hasta este punto habremos obtenido:

```
> Polbase<-function(n,t,s){
+
+ 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)
+ }
>
```

- Pasamos ahora a calcular la función del polinomio interpolador (polinterp), que dependerá de n, B (valores en y, dados por el problema) y de L (Calculado anteriormente). Así, nuestra función será "function(n,B,L)"
- Dentro de la función, igualamos p=0(*), y abrimos un bucle for, con la i variando desde 1 hasta n.
- Dentro del bucle hacemos la operación del sumatorio.
- Cerramos el bucle, y, de nuevo, antes de cerrar la función indicamos Return(p).

Así habremos hecho:

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

(*) al final p nos va a calcular el valor estimado del valor t que introducimos. Hasta ahora no se han introducido datos numéricos en el problema ni qué valores toma t, por lo que no obtendremos ningún resultado si lo ejecutamos.

- En este punto podemos introducir los datos del problema. Conviene tener en cuenta que n es la longitud del vector B o S.
- Ahora sí definimos lo que representan L y p. Para ello igualamos *L* a la función *Polbase* y *p* a polinterp. Ahora, al introducir cualquier valor de t obtendremos el valor estimado en y. (A la hora de trabajar con tantas variables y diferentes comandos, es frecuente cometer algunos errores, cuidado !! **Prestad especial atención al orden en el que introducir los argumentos de los que dependen la función polbase y polinterp, ya que tienen estar en el mismo orden cuando definis L y p al final**)

```
> B=c(20,25,32.46,56.3)
> s=c(6,7.2,8.23,9.77)
> n=length(B)
> t=6.5
>
> L=Polbase(n,t,s)
> L
[1]  0.39252239  0.89045559 -0.32355863  0.04058065
> p=PolInterp(n,B,L)
> p
[1] 21.89381
>
```

Aquí concluiría el ejercicio como tal, ya que ya hemos obtenido el valor en el punto t dado. Sin embargo, es posible que nos pidan representar ambas funciones, así que vamos a ver rápidamente cómo poder representar conjuntamente el polinomio interpolador y los puntos que nos dan.

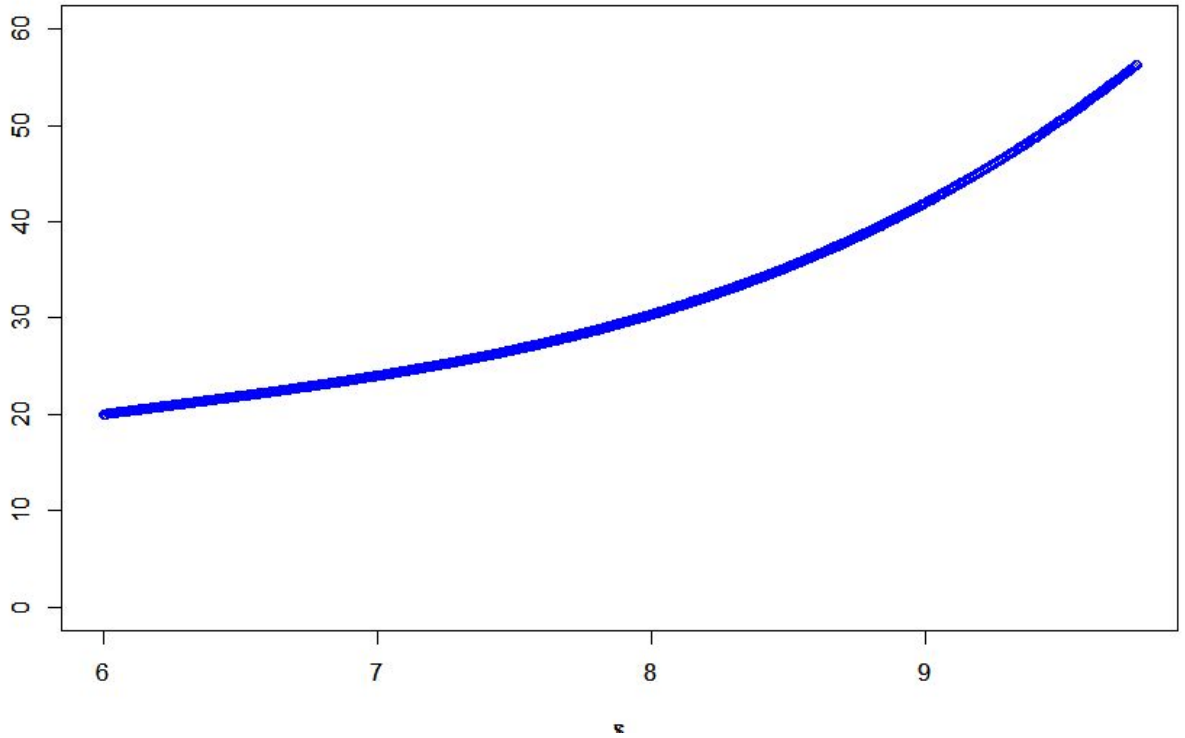
- El primer paso es generar un vector con numerosos valores de x, que luego nos servirán como valores de t (es decir, queremos varias componentes x de nuestra función interpoladora para poder representarla). (hay varios métodos para hacer esto, elegid el que queráis, en el ejercicio resuelto podéis ver otra manera)
- Creamos un vector f, que contenga los correspondientes puntos del vector x, calculados todos mediante el polinomio interpolador. Para hacer eso:
    - Creamos un bucle, en el que [k] varía desde 1 hasta el número de componentes que tenga nuestro vector x.
    - Dentro del bucle establecemos que t (que, recordad, es el punto que queremos calcular su componente y mediante el polinomio interpolador), va a ser igual a la componente [k] del vector x.
    - Volvemos a calcular todos los valores de L, a partir de los valores de t (x) que tenemos.
    - Finalmente, obtenemos el vector f (de k componentes), cuyas componentes f[k] serán iguales a los valores que tome el polinomio interpolador (Polinterp).

```
> x=seq(s[1],s[n],length.out=1001)
> f=0
> for (k in 1:1001){
+ t=x[k]
+ L=Polbase(n,t,s)
+ f[k]=PolInterp(n,B,L)
+ }
>
```

- Solo nos queda definir las 2 funciones con “plot()”, una de ellas contendrá los valores **(s,B)** y la otra los **(x,f).** Es importante que indiquemos que ambas funciones tienen que tener la misma escala, para lo que usaremos “ylim=c()” y “xlim=c()” para establecer los mismos límites. (Dentro de plot podemos incluir diversos comandos que lo modifican, se encuentran explicados en funciones plot).

```
> plot(s,B,pch=19,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))
> 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), col='blue')
>
```

La gráfica que obtendremos finalmente será algo similar a(*):



(*): Esta gráfica es específica de este ejemplo. Las gráficas pueden cambiar según el número de puntos que tengamos, ya que el grado del polinomio varía también con este.

Para finalizar os dejamos un enlace a un ejercicio resuelto que incluye una interpolación de Lagrange.