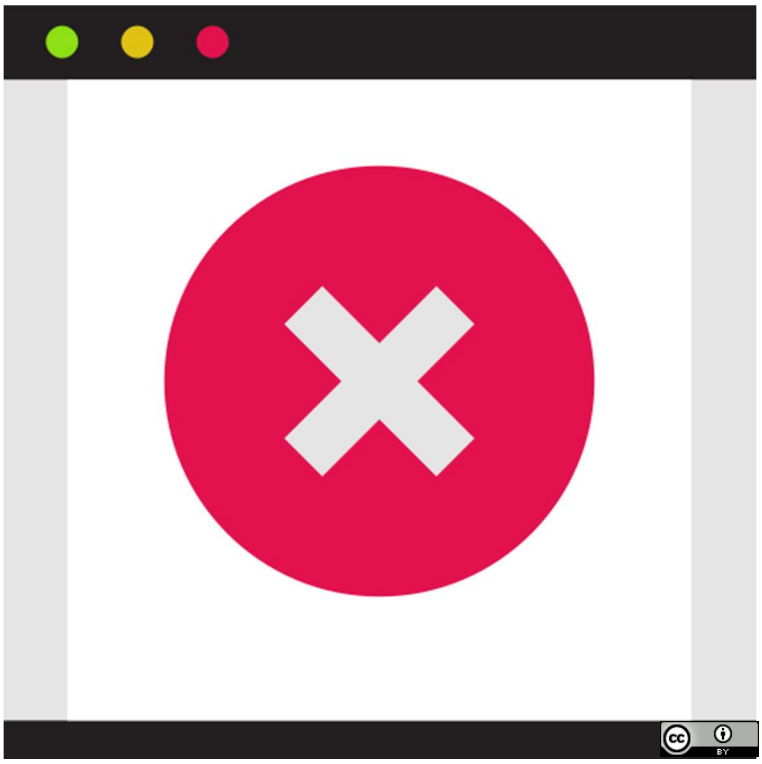
Fallos en bucles y estructuras condicionales



¿Cansado de que te aparezca un '+' cada vez que haces un bucle? ¿Quieres saber por qué no obtienes los resultados que deberías? Aquí podrás encontrar los errores más comunes en bucles y condicionales para que todo te salga de 2 (en binario).

Fallos en bucles y estructuras condicionales

- Debemos prestar especial atención al empleo de **paréntesis, llaves** y **corchetes**.

Si no escribimos paréntesis '( )' al crear un bucle aparecerá un error tipo *'unexpected symbol'*.

```
> n=3
> for j in 1:n
Error: unexpected symbol in "for j"
> for (j in 1:n){
+ }
```

En este tipo de errores debemos revisar lo que hemos escrito justo antes del símbolo mencionado por R.

```
> v=c(1,2,3,4)
> w=c(2,6,7,8)
> n=4
> for(i in 1:n)
+ z[i]=w[i]-v[i]
Error: objeto 'z' no encontrado
```

No podemos olvidar de abrir una llave '{ }' al crear un bucle, justo antes del proceso deseado. Si no lo escribimos, R tomará el proceso como una operación fuera del bucle y avisará de un error 'objeto no encontrado'.

Otro error ocurre al operar con vectores. Debemos emplear corchetes '[ ]', si no lo hacemos R nos avisará de que 'no se pudo encontrar la función'.

```
> v=c(1,2,3,4)
> w=c(2,6,7,8)
> n=4
> for(i in 1:n){
+ z(i)=w(i)-v(i)
+ }
Error in w(i) : no se pudo encontrar la función "w"
> v=c(1,2,3,4)
> w=c(2,6,7,8)
> n=4
> for(i in 1:n){
+ z[i]=w[i]-v[i]
+ }
```

- No debemos olvidarnos de darle **carácter vectorial** al resultado de una operación con vectores.

Si no le damos este carácter (mediante corchetes, '[ ]'), almacenará un único valor, ya que será tomado como variable.

Lo mismo sucede con las matrices.

Atención: R no nos va a avisar del error ya que desconoce el carácter que deseamos emplear.

```
> v=c(1,2,3,4)
> w=c(2,6,7,8)
> n=4
> for(i in 1:n){
+ z=w[i]-v[i]
+ }
> z
[1] 4
> v=c(1,2,3,4)
> w=c(2,6,7,8)
> n=4
> for(i in 1:n){
+ z[i]=w[i]-v[i]
+ }
> z
[1] 1 4 4 4
```

- Debemos asegurarnos de **cerrar** todos los **paréntesis**, **llaves** y **corchetes**.

```
> a=c(2,6,3,8)
> b=c(4,9,2,7)
> c=0
> for (i in 1:4){
+ c[i]=a[i]+b[i]
+ if(c<9){
+ print('L')
+ else{
Error: inesperado 'else' in:
"print('L')
else"
```

Si no lo hacemos, y este fallo ocurre en mitad del proceso, aparecerá un error 'inesperado'.

Si ocurre al final del proceso, aparecerá un signo '+' a la izquierda, lo que significa que debemos añadir algún símbolo.

```
> a=c(2,6,3,8)
> b=c(4,9,2,7)
> c=0
> for (i in 1:4){
+ c[i]=a[i]+b[i]
+ if(c<9){
+ print('L')}
+ else{
+ print('B')
+ }
+
```

- Para establecer **varias condiciones** en un mismo bucle condicional sólo podemos emplear el símbolo **'&'**.

```
> a=c(2,6,3,8); b=c(5,9,2,7)
> c=0
> for (i in 1:4){
+ c[i]=a[i]+b[i]
+ if(c[i]<9,c[i]>6){
Error: inesperado ',' in:
"c[i]=a[i]+b[i]
if(c[i]<9,"
```

Si utilizamos otras expresiones como 'and' o separamos las condiciones con comas, aparecerá un error 'inesperado' o *'unexpected symbol'*, respectivamente.

```
> for (i in 1:4){
+ c[i]=a[i]+b[i]
+ if(c[i]<9 and c[i]>6){
Error: unexpected symbol in:
"c[i]=a[i]+b[i]
if(c[i]<9 and"
```

```
> a=c(2,6,3,8); b=c(5,9,2,7)
> c=0
> for (i in 1:4){
+ c[i]=a[i]+b[i]
+ if(c[i]<9 & c[i]>6){
+ print('L')}
+ else{
+ print('B')}}
[1] "L"
[1] "B"
[1] "B"
[1] "B"
```

- Debemos especificar el **carácter vectorial o matricial** de un elemento al escribir la condición (v[k], no v).

```
> a=c(2,6,3,8); b=c(5,9,2,7)
> c=0
> for (i in 1:4){
+ c[i]=a[i]+b[i]
+ if(c<9 & c>6){
+ print('L')}
+ else{
+ print('B')}}
[1] "L"
[1] "L"
[1] "L"
[1] "L"
Warning messages:
1: In if (c < 9 & c > 6) { :
  la condición tiene longitud > 1 y sólo el primer elemento será usado
2: In if (c < 9 & c > 6) { :
  la condición tiene longitud > 1 y sólo el primer elemento será usado
3: In if (c < 9 & c > 6) { :
  la condición tiene longitud > 1 y sólo el primer elemento será usado
```

```
> a=c(2,6,3,8); b=c(5,9,2,7)
> c=0
> for (i in 1:4){
+ c[i]=a[i]+b[i]
+ if(c[i]<9 & c[i]>6){
+ print('L')}
+ else{
+ print('B')}}
[1] "L"
[1] "B"
[1] "B"
[1] "B"
```

Aparecerá un aviso (*'warning message'*) en el que R nos informa de que sólo va a emplear el primer elemento (todas las soluciones serán iguales).

- Debemos tener cuidado al escribir el **proceso** de un **bucle 'while'**.

```
> d<-function(x){
+ exp(x)*sin(x)
+ }
> A=0; B=3.05; n=35
> h=(B-A)/(n-1)
> T=seq(A,B,h)
> T
 [1] 0.00000000 0.08970588 0.17941176 0.26911765 0.35882353 0.44852941
 [7] 0.53823529 0.62794118 0.71764706 0.80735294 0.89705882 0.98676471
[13] 1.07647059 1.16617647 1.25588235 1.34558824 1.43529412 1.52500000
[19] 1.61470588 1.70441176 1.79411765 1.88382353 1.97352941 2.06323529
[25] 2.15294118 2.24264706 2.33235294 2.42205882 2.51176471 2.60147059
[31] 2.69117647 2.78088235 2.87058824 2.96029412 3.05000000
> Sl=0
> i=2
> while(i<=(n-1)){
+ Sl=Sl+d(T[i])
+ }
```

Si mantenemos la misma variable el bucle puede realizarse infinitas veces y bloquear el programa.

Debemos asegurarnos de que la variable va a ir cambiando a lo largo del proceso empleando estructuras como 'k=k+1'.

```
> Sl=0
> i=2
> while(i<=(n-1)){
+ Sl=Sl+d(T[i])
+ i=i+1
+ }
> Sl
[1] 132.4214
```