Condicionales y bucles

Angela Di Serio

Contents

Operadores Lógicos **1**

Estructuras Condicionales **2**

IF 2

Ejemplo 2

FOR 3

while 3

break y next 4

Operadores Lógicos

Los operadores lógicos son usados para operaciones de álgebra Booleana, es decir, para describir relaciones lógicas, expresadas como verdadero (TRUE) o falso (FALSE).

Operador	Comparación	Ejemplo	Resultado
`x | y`	x Ó y es verdadero	`TRUE | FALSE`	`TRUE`
`x & y`	x Y y son verdaderos	`TRUE & FALSE`	`FALSE`
`!x`	x no es verdadero (negación)	`!TRUE`	`FALSE`
`isTRUE(x)`	x es verdadero (afirmación)	`isTRUE(TRUE)`	`TRUE`

Los operadores `|` y `&` siguen estas reglas:

- `|` devuelve `TRUE` si alguno de los datos es `TRUE`
- `&` solo devuelve `TRUE` si ambos datos es `TRUE`
- `|` solo devuelve `FALSE` si ambos datos son `FALSE`
- `&` devuelve `FALSE` si alguno de los datos es `FALSE`

Estructuras Condicionales

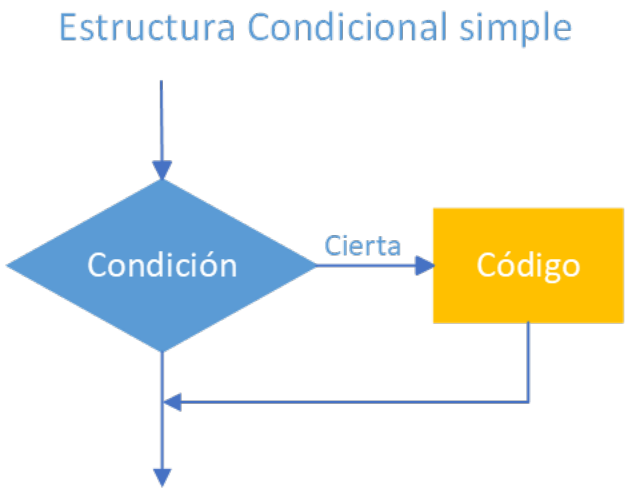
IF

Las estructuras de control condicionales permiten que un programa decida de manera automática entre varias opciones en función de si se cumplen o no determinadas condiciones.

Estas estructuras tienen en R la misma estructura que en casi todos los otros lenguajes de programación.

La estructura que se muestra a continuación:

```
if (condición) {código}
```



le indica a R que si se satisface la **condición** especificada entre paréntesis, se lleve a cabo el **código** indicado entre llaves y luego se continúa con el programa, mientras que si no se satisface la condición se continúa con el programa sin efectuar el **código**.

Por otro lado, tenemos la estructura:

```
if (condición) {SI_acciones} else {NO_acciones}
```

que indica a R que si se satisface la **condición** se lleve a cabo las acciones **SI_acciones** y si no se satisfe dicha condición se lleven a cabo las acciones **NO_acciones**. Luego en ambos casos se continua con el programa.

Ejemplo

```
if (n>10 & n< 50) {
  print("n es mayor que 10 y menor que 50")
} else {
  print("n no cumple con la condición indicada")
}
```

FOR

La estructura **for** nos permite ejecutar un bucle, realizando una operación para cada elemento de un conjunto de datos.

Su estructura es la siguiente:

```
for (elemento in objeto) {
  operacion_con_elemento
}
```

Para cada **elemento** en el **objeto** debe realizar **operacion_con_elemento**.

```
objeto = 1:10
suma = 0
for (elemento in objeto) {
  suma = suma + elemento
}
print(suma)
```

```
## [1] 55
```

```
dado = 1:6
mi_vector=vector()
for(cara in dado) {
  mi_vector[cara] <- cara ^ 2
}
print(mi_vector)
```

```
## [1]  1  4  9 16 25 36
```

En R generalmente hay opciones mejores, en cuanto a simplicidad y velocidad de cómputo, que un bucle for. El ejemplo anterior es preferible hacerlo como:

```
dado = 1:6
mi_vector = dado ^ 2
print(mi_vector)
```

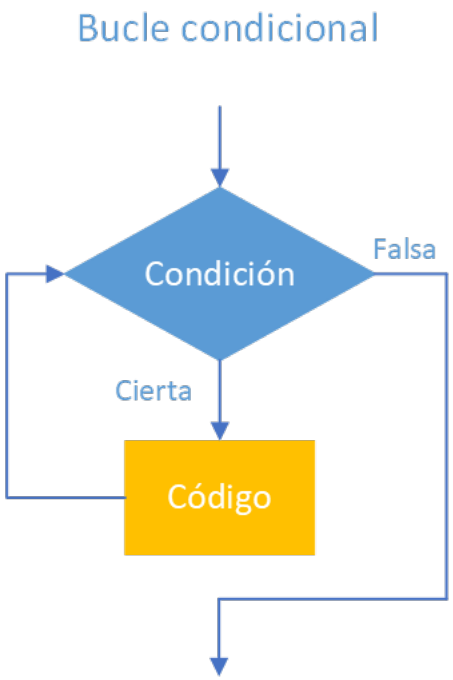
```
## [1]  1  4  9 16 25 36
```

while

Este es un tipo de bucle que ocurre mientras una **condición** es verdadera (TRUE). La operación se realiza hasta que se se llega a cumplir un criterio previamente establecido.

El modelo de while es:

```
while (condición) {
  código
}
```




```
umbral = 5
valor  = 0

while(valor < umbral) {
  print("Todavía no.")
  valor = valor + 1
}
```

```
## [1] "Todavía no."
## [1] "Todavía no."
## [1] "Todavía no."
## [1] "Todavía no."
## [1] "Todavía no."
```

```
print("Ahora si.")
```

```
## [1] "Ahora si."
```

break y next

break y next son palabras reservadas en R, no podemos asignarles nuevos valores y realizan una operación específica cuando aparecen en nuestro código.

break nos permite interrumpir un bucle, mientras que next nos deja avanzar a la siguiente iteración del bucle, "saltándose" la actual. Ambas funcionan para for y while.

```r
for(i in 1:10) {
  if(i == 3) {
    break
  }
  print(i)
}
```

```
## [1] 1
## [1] 2
```

```r
numero <- 20

while(numero > 5) {
  if(numero == 15) {
    break
  }
  numero <- numero - 1
}

numero
```

```
## [1] 15
```