Condiciones Lógicas

En este recurso se van a explicar las bases del álgebra booleana que se usan en programación y cómo se pueden aplicar en condiciones y bucles. Además, se incluirán dos ejercicios a modo de ejemplo para ayudar a afianzar los conceptos explicados.

Conceptos Básicos del Álgebra Booleana

El álgebra booleana se basa en un sistema de operaciones lógicas de variables que solo pueden tomar dos valores: verdadero (1, TRUE) o falso (0, FALSE). Hay tres operaciones básicas que pueden representarse en tablas de verdad.

- **Operador AND (&):** es la operación "producto", todas las variables deben ser verdaderas para que la conjunción sea verdadera.

A	B	A&B
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

- **Operador OR (|):** es la operación "suma" y basta con que una variable sea verdadera para que el conjunto de todas lo sea.

A	B	A|B
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

- **Operador NOT (!):** esta operación niega la variable y devuelve su complemento, es decir, si es verdadero devolverá falso y viceversa.

A	!A
0	**1**
1	**0**

Operadores Lógicos y Relacionales en R

En R hay dos tipos de operadores que devuelven los valores TRUE o FALSE. Por un lado, tenemos los operadores lógicos que se corresponden con los operadores explicados en el apartado anterior. Estos nos sirven para comparar dos variables que tengan asignados valores lógicos. Por otro lado, los operadores relacionales sirven para comparar objetos, generalmente de valor numérico. Estas operaciones devuleven TRUE o FALSE como resultado y pueden combinarse con los lógicos.

- **Operadores lógicos:**

&	AND
&&	AND
|	OR
||	OR
!	NOT

Podemos observar que hay dos operadores para AND y otros dos para OR. Aunque llevan a cabo el mismo tipo de comparación, se diferencian en que al utilizar un único símbolo la comparación se hace elemento a elemento, mientras que si se utilizan dos R solo compara un elemento y da error si hay más. Es decir, en el primer caso se pueden comparar tanto vectores como dos elementos aislados, pero el doble símbolo solo permite hacer esto último. Por tanto, es recomendable el uso de un único símbolo para evitar errores y confusiones ya que la operación y el resultado final son iguales.

- **Operadores relacionales:**

==	Igual a
!=	Distinto de
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

Estos operadores sirven para comparar cualquier tipo de dato. Lo más común es comparar datos de tipo numérico, a lo que R devolverá TRUE o FALSE dependiendo de si se cumple la condición matemática asignada. Aunque también se pueden comparar datos de tipo string por orden alfabético, no es de mucha utilidad en esta asignatura por lo que no se va a demostrar su uso.

- **Ejemplos de uso:**

```
> # Operadores lógicos
> a = TRUE
> b = FALSE
> c = TRUE
> d = FALSE
>
> a & b # AND devuelve FALSE porque cada variable tiene un valor distinto
[1] FALSE
> a | b | d # OR devuelve TRUE porque al menos una variable es verdadera
[1] TRUE
> !b # NOT el opuesto de FALSE es TRUE
[1] TRUE
> !a & c # AND devuelve FALSE por la negación NOT
[1] FALSE
> (a | d) & c # AND devuelve TRUE porque la condición OR entre paréntesis lo es
[1] TRUE
> (a & c) | (b & d) # OR devuelve TRUE porque al menos uno de los paréntesis es verdadero
[1] TRUE
> (a | c) & (b | d) # AND devuelve FALSE porque los dos paréntesis no son verdaderos
[1] FALSE
```

```
> # Operadores relacionales
> a = 2
> b = 3
> c = 8
> d = 5
>
> a == a # TRUE la variable es igual a sí misma
[1] TRUE
> b == d # FALSE las variables tienen valores distintos
[1] FALSE
> c != d # TRUE las variables tienen valores distintos
[1] TRUE
> a >= c # FALSE a no es mayor o igual que c
[1] FALSE
> b <= b # TRUE b es menor o igual que b
[1] TRUE
> d < c # TRUE d es menor que c
[1] TRUE
> b > d # FALSE b no es mayor que d
[1] FALSE
> (a + b) == d # TRUE d es la suma de a y b
[1] TRUE
> (c - d) < b # FALSE la resta de c menos d no es menor que b
[1] FALSE
> (c - a) != 2*b # FALSE ambas operaciones dan el mismo resultado
[1] FALSE
```

```
> # Operadores lógicos y relacionales
> a = TRUE
> b = FALSE
> c = TRUE
> d = FALSE
> t = 2
> u = 3
> v = 8
> w = 5
>
> b == d # TRUE ambas variables son falsas y por tanto iguales
[1] TRUE
> (t + u == w) & a # TRUE ambas expresiones son verdaderas
[1] TRUE
> (t != v) & (!c == d) # TRUE ambas expresiones son verdaderas
[1] TRUE
> ((v - u == w) | !a) & ((t > v) | (b != c)) # TRUE se cumple una de las concidiones a cada lado
[1] TRUE
```

Ejercicios con bucles

En R, como en otros lenguajes de programación, los bucles while e if aceptan condiciones lógicas. Si la condición es verdadera (TRUE) se realizará lo que esté indicado dentro del bucle. Para entender esto mejor, aquí hay unos ejemplos simples que ayudarán a comprender el uso de las condiciones en los bucles.

Ejercicio 1: determinar si un número introducido por el usuario es mayor o igual que 3 y a su vez es menor que 10.

```
num = as.numeric(readline("Introduce un número: "))

if (num>=3 & num<10){
  cat(num, "es mayor o igual que 3 y menor que 10", "\n")
} else if (num<3){
  cat(num, "es menor que 3", "\n")
} else if (num>=10){
  cat(num, "es mayor o igual que 10", "\n")
}
```

```
Introduce un número: 5
5 es mayor o igual que 3 y menor que 10
Introduce un número: 10
10 es mayor o igual que 10
Introduce un número: 1
1 es menor que 3
```

Ejercicio 2: a partir de un vector de números enteros, crear un vector que contenga los números pares y otro que contenga los impares.

```
numeros = c(1, 6, 3, 8, 23, 5, 9, 10, 34, 12)
pares = c()
impares = c()
i = 1
while (i <= length(numeros)){
  if (numeros[i]%%2 == 0){
    pares = c(pares, numeros[i])
  } else if (numeros[i]%%2 != 0){
    impares = c(impares, numeros[i])
  }
  i = i + 1
}
```

Nota: en este ejercicio, el bucle for es más recomendable que el while, ya que se conoce la longitud del vector y por tanto cuando debe acabar, pero se ha usado un bucle while para enseñar el uso de las condiciones en este tipo de bucles.