Operadores lógicos e instrucciones condicionales

Anteriormente, aprendiste que un **operador** es un símbolo que nombra al tipo de operación o cálculo que se realizará en una fórmula. En esta lectura, aprenderás sobre los principales tipos de operadores lógicos en R y cómo se pueden utilizar para crear instrucciones condicionales en código R.

# Operadores lógicos

Los **operadores lógicos** arrojan un tipo de dato lógico, por ejemplo, TRUE o FALSE.

Existen tres tipos principales de operadores lógicos:

* AND (algunas veces representado como & o && en R)
* OR (algunas veces representado como | o || en R)
* NOT (!)

Esta tabla resume los operadores lógicos:

|  |  |  |
| --- | --- | --- |
| **El operador AND (“&”)** | **Operador OR (“|”)** | **El operador NOT (“!”)** |
| El operador AND requiere de dos valores lógicos. El resultado es TRUE si *cada uno* de los valores por separado es TRUE. Esto significa que TRUE & TRUE arroja como resultado TRUE.Sin embargo, FALSE & TRUE, TRUE & FALSE y FALSE & FALSE arrojan como resultado FALSE. | El operador OR (|) trabaja de un modo similar el operador AND (&). La diferencia principal consiste en que, al menos, uno de los valores de la operación OR debe ser TRUE para que toda la operación OR dé como resultado TRUE. | El operador NOT (!) simplemente niega el valor lógico al que se aplica. En otras palabras, !TRUE arroja como resultado FALSE y !FALSE arroja como resultado TRUE. |
| Esto significa que TRUE | TRUE, TRUE | FALSE y FALSE | TRUE arrojan como resultado TRUE. Cuando ambos valores son FALSE, el resultado es FALSE. |
| Si tú ejecutas el código correspondiente en R, obtienes los siguientes resultados: | Si escribes el código, obtienes los siguientes resultados: | Cuando ejecutas el código, obtienes los siguientes resultados: |
| > TRUE & TRUE | > TRUE | TRUE | > !TRUE |
| [1] TRUE | [1] TRUE | [1] FALSE |
| > TRUE & FALSE | > TRUE | FALSE | > !FALSE |
| [1] FALSE | [1] TRUE | [1] TRUE |
| > FALSE & TRUE | > FALSE | TRUE |  |
| [1] FALSE | [1] TRUE | Al igual que los operadores OR y AND, puedes usar el operador NOT en combinación con operadores lógicos. Cero se considera FALSE y los números que no son cero se consideran |
| > FALSE & FALSE | > FALSE | FALSE |
| [1] FALSE | [1] FALSE |

|  |  |  |
| --- | --- | --- |
| Puedes ilustrar esto usando los resultados de nuestras comparaciones. Imagina que creas una variable *x* que equivale a 10.  x <- 10  Para ver si “x” es mayor que 3 pero menor que 12, puedes usar x > 3 y x < 12 como los valores de una expresión “AND”.  x > 3 & x < 12  Cuando se ejecuta la función, R arroja el resultado TRUE  [1] TRUE  La primera parte, x > 3 arrojará el resultado TRUE dado que 10 es mayor que 3. La segunda parte, x < 12, también dará como resultado TRUE, ya que 10 es menor que 12. Como *ambos* valores dan TRUE, el resultado de la expresión AND es TRUE. El número 10 se ubica entre los números 3 y 12.  Sin embargo, si haces que “x” sea igual a 20, la expresión x > 3 & x < 12 dará un resultado diferente. | Por ejemplo, supón que creas una variable *y* que es igual a 7. Para ver si *y* es menor que 8 o mayor que 16, puedes usar la siguiente expresión:  y <- 7  y < 8 | y > 16  El resultado de la comparación es TRUE (7 es menor que 8) | FALSE (7 no es mayor que 16). Ya que un solo valor de una expresión OR debe ser TRUE para que toda la expresión sea TRUE, R arroja como resultado TRUE.  [1] TRUE  Ahora imagina que y es 12. La expresión y < 8| y > 16 ahora da FALSE (12 < 8) | FALSE (12 > 16). Ambas comparaciones dan como resultado FALSE, de modo que el resultado es FALSE.  y <- 12  y < 8 | y > 16  [1] FALSE | como TRUE. El operador NOT da como resultado el valor lógico opuesto.  Imaginemos que tienes una variable "x" que equivale a 2:  x <- 2  La operación NOT da FALSE porque toma el valor lógico opuesto a un número que no es cero (TRUE).  > !x  [1] FALSE |

|  |  |  |
| --- | --- | --- |
| x <- 20  x > 3 & x < 12  [1] FALSE  Si bien x > 3 es TRUE (20 > 3), x < 12 es FALSE (20 < 12). Una parte de la expresión AND es FALSE, entonces, toda la expresión dará como resultado FALSE (TRUE & FALSE = FALSE). De modo que el resultado de R será FALSE. |  |  |

Ahora veamos un ejemplo de cómo puedes utilizar operadores lógicos para analizar los datos. Imagina que estás trabajando con un conjunto de datos “airquality” que está precargado en RStudio. Contiene datos sobre las mediciones de la calidad del aire en Nueva York desde mayo hasta septiembre de 1973.

El marco de datos tiene seis columnas: Ozone (la medición de ozono), Solar.R (la medición solar), Wind (la medición del viento), Temp (la temperatura en grados Fahrenheit), y Month y Day de esas mediciones (cada fila representa una combinación específica de mes y día).

Ahora veamos cómo los operadores AND, OR y NOT pueden ser de ayuda en esta situación.

**Ejemplo del operador AND**

Imagina que quieres especificar las filas que son extremadamente soleadas y ventosas, que defines como con una medición Solar de más de 150 *y* una medición de Wind de más de 10.

En R, puedes expresar esta instrucción lógica como Solar.R > 150 & Wind > 10. Solo las filas donde *ambas* condiciones son verdaderas cumplen con los criterios:

**Ejemplo del operador OR**

Luego, imagina que quieres especificar las filas que son extremadamente soleadas o ventosas, que defines como con una medición Solar de más de 150 *o* una medición de Wind de más de 10.

En R, puedes expresar esta instrucción lógica como Solar.R > 150 | Wind > 10. Todas las filas donde *una* de estascondiciones es verdadera cumplen con los criterios:

**Ejemplo del operador NOT**

Ahora, imagina que quieres concentrarte en las mediciones del clima de días que *no* son el primer día del mes. En R, puedes expresar esta instrucción lógica como Día != 1.

Las filas donde esta condición es verdadera cumplen con el criterio:

Para finalizar, imagina que quieres concentrarte en escenarios que no son extremadamente soleados ni ventosos, basándote en tus definiciones previas de extremadamente soleado y ventoso. En otras palabras, la siguiente instrucción *no* debería ser verdadera: ya sea que haya una medición Solar mayor que 150 *o* una medición de Wind mayor que 10.

Observa que esta instrucción es la opuesta de la instrucción OR utilizada anteriormente. Para expresar esta instrucción en R, puedes colocar un signo de exclamación (!) frente a la instrucción OR previa: !(Solar.R > 150 | Viento > 10). R aplicará el operador NOT a todo lo que se encuentre entre paréntesis.

Es este caso, solo una fila cumple con el criterio.

# Instrucciones condicionales

Una **instrucción condicional** es una declaración de que si una determinada condición se mantiene, entonces, debe producirse un determinado evento. Por ejemplo, "*Si* la temperatura está por encima del punto de congelamiento, *entonces,* saldré a caminar". Si la primera condición es verdadera (la temperatura está por encima del punto de congelamiento), entonces, la segunda condición sucederá (salir a caminar). Las instrucciones condicionales en código R tienen una lógica similar.

Veamos ahora cómo crear instrucciones condicionales en R utilizando tres instrucciones relacionadas:

* **if()**
* **else()**
* **else if()**

## Instrucción if

La instrucción **if** establece una condición y si la condición arroja como resultado TRUE, el código R asociado a la instrucción if se ejecuta.

En R, escribes el código para la condición entre los paréntesis de la instrucción if. El código que debe ser ejecutado si la condición da TRUE se escribe entre llaves (“expr”). Observa que, en este caso, las segundas llaves se colocan en la propia línea del código e identifican el final del código que quieres ejecutar.

if (condition) { expr

}

Por ejemplo, creemos una variable “x” igual a 4.

x <- 4

Luego, creemos una instrucción condicional: si x es mayor que 0, entonces, R imprimirá la cadena "x es un número positivo".

if (x > 0) {

print("x is a positive number")

}

Dado que x=4, la condición es verdadera (4 > 0). Por lo tanto, cuando ejecutas el código, R imprime la cadena "x es un número positivo".

[1] "x is a positive number"

Pero si cambias x a un número negativo, como -4, entonces, el resultado de la condición será FALSE (-4 > 0). Si ejecutas el código, R no ejecutará la instrucción impresa. En su lugar, aparecerá una línea en blanco.

## Instrucción else

La instrucción **else** se utiliza en combinación con una instrucción **if**. Así se estructura el código en R:

if (condition) {

expr1

} else {

expr2

}

El código asociado a la instrucción else se ejecuta cada vez que la condición de la instrucción if *no sea* TRUE. En otras palabras, si la condición es TRUE, entonces, R ejecutará el código en la instrucción if (“expr1”); si la condición *no* esTRUE, entonces, R ejecutará el código else en la instrucción else (“expr2”).

Veamos un ejemplo. Primero, creemos una variable “x” igual a 7.

x <- 7

Ahora, establezcamos las siguientes condiciones:

* Si x es mayor que 0, R imprimirá "x es un número positivo".
* Si x es menor o igual a 0, R imprimirá "x es un número negativo o cero".

En nuestro código, la primera condición (x > 0) será parte de la instrucción if. La segunda condición de x menor o igual a 0 queda implícita en la instrucción else. Si x > 0, R imprimirá "x es un número positivo". De otro modo, R imprimirá "x es un número negativo o cero".

x <- 7

if (x > 0) {

print("x is a positive number")

} else {

print ("x is either a negative number or zero")

}

Dado que 7 es mayor que 0, la condición de la instrucción if es verdadera. Por lo tanto, cuando ejecutas el código, R imprime "x es un número positivo".

[1] "x is a positive number"

Sin embargo, si haces que x sea igual a -7, la condición de la instrucción if *no* será verdadera (-7 no es mayor que 0). Por lo tanto, R ejecutará el código en la instrucción else. Cuando ejecutas el código, R imprime "x es un número positivo o cero".

x <- -7

if (x > 0) {

print("x is a positive number")

} else {

print ("x is either a negative number or zero")

}

[1] "x is either a negative number or zero"

## Instrucción else if

En algunos casos, podrías querer personalizar aún más tu instrucción condicional agregando una instrucción **else if**. La instrucción else if se ubica entre la instrucción if y la instrucción else.

Esta es la estructura del código:

if (condition1) {

expr1

} else if (condition2) {

expr2

} else {

expr3

}

Si la condición if (“condition1”) se cumple, entonces R ejecuta el código en la primera expresión (“expr1”). Si la condición if no se cumple, y la condición else if (“condition2”) se cumple, entonces, R ejecuta el código en la segunda expresión (“expr2”). Si ninguna de las dos condiciones se cumple, R ejecuta el código en la tercera expresión (“expr3”).

En nuestro ejemplo anterior, utilizando solo las instrucciones if y else, R puede solo imprimir "x es un número negativo o cero" si x es igual a 0 o x es menor que 0. Imagina que quieres que R imprima la cadena "X es cero" si x es igual a 0. Necesitas agregar otra condición utilizando la instrucción else if.

Veamos un ejemplo. Primero creemos una variable “x” igual a 1 negativo ("-1").

x <- -1

Ahora, quieres establecer las siguientes condiciones:

* Si x es menor que 0, imprimir "x es un número negativo".
* Si x es igual a 0, imprimir " x es cero".
* De otro modo, imprimir "x es un número positivo".

En nuestro código, la primera condición será parte de la instrucción if, la segunda condición será parte de la instrucción else if y la tercera condición será parte de la instrucción else. Si x < 0, R imprimirá "x es un número positivo". Si x = 0, R imprimirá "x es cero". De otro modo, R imprimirá "x es un número positivo".

x <- -1

if (x < 0) {

print("x is a negative number")

} else if (x == 0) {

print("x es cero")

} else {

print("x is a positive number")

}

Dado que -1 es menor que 0, la condición de la instrucción if resulta TRUE y R imprime "x es un número negativo".

[1] "x is a negative number"

Si haces que x sea igual a 0, R comprobará primero la condición if (x < 0) y determinará que es FALSE. Entonces, R evaluará la condición else if. Esta condición, x==0, es TRUE. Por ello, en este caso, R imprimirá "x es cero".

Si haces que x sea igual a 1, ambas condiciones, if y else, arrojarán como resultado FALSE. De modo que R ejecutará la instrucción else e imprimirá "x es un número positivo".

Tan pronto como R descubre una condición que resulta TRUE, R ejecuta el código correspondiente e ignora el resto.

# Recursos

Para saber más acerca de las operaciones lógicas y las instrucciones condicionales, consulta el tutorial sobre [“Condicionales y flujo de control en R” en [el sitio web de](https://www.datacamp.com/community/tutorials/conditionals-and-control-flow-in-r) DataCamp. DataCamp es un recurso popular entre quienes estudian programación informática. El tutorial abunda en ejemplos útiles de aplicaciones de codificación para operadores lógicos e instrucciones condicionales (y operadores relacionales) y ofrece un buen panorama de cada tema y las conexiones entre ellos.](https://www.datacamp.com/community/tutorials/conditionals-and-control-flow-in-r)