R operators

You might remember that an **operator** is a symbol that identifies the type of operation or calculation to be performed in a formula. In an earlier video, you learned how to use the assignment and arithmetic operators to assign variables and perform calculations. In this reading, you will get a detailed summary of the main types of operators in R, and learn how to use specific operators in R code.

Operators

In R, there are four main types of operators:

1. Arithmetic
2. Relational
3. Logical
4. Assignment

Let's review the specific operators in each category and check out some examples of how to use them in R code.

Arithmetic operators

Arithmetic operators let you perform basic math operations like addition, subtraction, multiplication, and division.

The table below summarizes the different arithmetic operators in R. The examples used in the table are based on the creation of two variables: : *x* equals 2 and *y* equals 5. Note that you use the assignment operator to store these values:

x <- 2

y <- 5

Operator	Description	Example Code	Result/ Output
+	Addition	x + y	[1] 7
-	Subtraction	x - y	[1] -3
*	Multiplication	x * y	[1] 10
/	Division	x / y	[1] 0.4
%%	Modulus (returns the remainder after division)	y %% x	[1] 1
%/%	Integer division (returns an integer value after division)	y%/% x	[1] 2
^	Exponent	y ^ x	[1]25

Relational operators

Relational operators, also known as comparators, allow you to compare values. Relational operators identify how one R object relates to another—like whether an object is less than, equal to, or greater than another object. The output for relational operators is either TRUE or FALSE (which is a logical data type, or boolean).

The table below summarizes the six relational operators in R. The examples used in the table are based on the creation of two variables: *x* equals 2 and *y* equals 5. Note that you use the assignment operator to store these values.

x <- 2

y <- 5

If you perform calculations with each operator, you get the following results. In this case, the output is boolean: TRUE or FALSE. Note that the [1] that appears before each output is used to represent how output is displayed in RStudio.

Operator	Description	Example Code	Result/Output
<	Less than	x < y	[1] TRUE
>	Greater than	x > y	[1] FALSE
<=	Less than or equal to	x < = 2	[1] TRUE
>=	Greater than or equal to	y >= 10	[1] FALSE
=	Equal to	y == 5	[1] TRUE
!=	Not equal to	x != 2	[1] FALSE

Logical operators

Logical operators allow you to combine logical values. Logical operators return a logical data type or boolean (TRUE or FALSE)**.** As a refresher, we encountered logical operators in an earlier reading, "Logical Operators and Conditional Statements," but let's review them again.

The table below summarizes the logical operators in R.

Operator	**Description**
&	Element-wise logical AND
&&	Logical AND
|	Element-wise logical OR
||	Logical OR
!	Logical NOT

Let's check out some examples of how logical operators work in R code.

Element-wise logical AND (&) and OR (|)

You can illustrate logical AND (&) and OR (|) by comparing numerical values. Let's create a variable *x* that is equal to 10.

```
x <- 10
```

The AND operator returns TRUE only if *both* individual values are TRUE.

```
x > 2 & x < 12
```

```
[1] TRUE
```

10 is greater than 2 *and* 10 is less than 12. So, the operation evaluates to *TRUE*.

The OR operator (|) works in a similar way to the AND operator (&). The main difference is that just *one* of the values of the OR operation needs to be TRUE for the entire OR operation to evaluate to TRUE. Only if *both* values are FALSE will the entire OR operation evaluate to *FALSE*.

Let's try an example with the same variable *(x <- 10)*:

```
x > 2 | x < 8
```

```
[1] TRUE
```

10 is greater than 2, but 10 is not less than 8. But since at least one of the values (10>2) is TRUE, the OR operation evaluates to *TRUE*.

Logical AND (&&) and OR (||)

The main difference between element-wise logical operators (&, |) and logical operators (&&, ||) is the way they apply to operations with vectors. The operations with double signs, AND (&&) and logical OR (||), only examine the *first* element of each vector. The operations with single signs, AND (&) and OR (|), examine all the elements of each vector.

For example, imagine you are working with two vectors that each contain three elements: *c(3, 5, 7)* and *c(2, 4, 6)*. The logical AND (&) will compare the first element of the first vector with the first element of the second vector (3&2), the second element with the second element (5&4), and the third element with the third element (7&6).

Let's check out this example in R code.

First, create two variables, *x* and *y*, to store the two vectors:

```
x <- c(3, 5, 7)

y <- c(2, 4, 6)
```

Then run the code with a single ampersand (&). The output is boolean (TRUE or FALSE).

```
x < 5 & y < 5

[1]  TRUE FALSE FALSE
```

When you compare each element of the two vectors, the output is *TRUE, FALSE, FALSE*. The first element of both *x* (3) and *y* (2) is less than 5, so this is TRUE. The second element of x is *not* less than 5 (it's equal to 5) but the second element of y is less than 5, so this is FALSE (because we used AND). The third element of both x and y is not less than 5, so this is also FALSE.

Now, let's run the same operation using the double ampersand (&&):

```
x < 5 && y < 5

[1] TRUE
```

In this case, R only compares the *first* elements of each vector: 3 and 2. So, the output is *TRUE* because 3 and 2 are both less than 5.

Depending on the type of work you do, you might make use of single sign operators more often than double sign operators. But it is helpful to know how all of the operators work regardless.

Logical NOT (!)

The NOT operator simply negates the logical value, and evaluates to its opposite. In R, zero is considered FALSE and all non-zero numbers are considered TRUE.

For example, let's apply the NOT operator to our variable *(x <- 10)*:

!(x < 15)

[1] FALSE

The NOT operation evaluates to *FALSE* because it takes the opposite logical value of the statement *x < 15*, which is TRUE (10 is less than 15).

Assignment operators

Assignment operators let you assign values to variables.

In many scripting programming languages you can just use the equal sign (=) to assign a variable. For R, the best practice is to use the arrow assignment (<-). Technically, the single arrow assignment can be used in the left or right direction. But the rightward assignment is not generally used in R code.

You can also use the double arrow assignment, known as a scoping assignment. But the scoping assignment is for advanced R users, so you won't learn about it in this reading.

The table below summarizes the assignment operators and example code in R. Notice that the output for each variable is its assigned value.

Operator	**Description**	**Example Code (after the sample code below, typing x will generate the output in the next column)**	**Result/ Output**
<-	Leftwards assignment	x < - 2	[1] 2
<<-	Leftwards assignment	x <<- 7	[1] 7
=	Leftwards assignment	x = 9	[1] 9
->	Rightwards assignment	11 -> x	[1] 11
->>	Rightwards assignment	21 - >> x	[1] 21

The operators you learned about in this reading are a great foundation for using operators in R.

Additional resource

Check out the article about R Operators on the R Coder website for a comprehensive guide to the different types of operators in R. The article includes lots of useful coding examples, and information about miscellaneous operators, the infix operator, and the pipe operator.