R Recursion (Recursive Function) With Example

R Recursion (Recursive Function) With Example

In this tutorial, you will learn to create a recursive function (a function that calls itself) in R programming.

A function that calls itself is called a recursive function and this technique is known as recursion.

This special programming technique can be used to solve problems by breaking them into smaller and simpler sub-problems.

An example can help clarify this concept.

Let us take the example of finding the factorial of a number. Factorial of a positive integer number is defined as the product of all the integers from 1 to that number. For example, the factorial of 5 (denoted as `5!`) will be

```
5! = 1*2*3*4*5 = 120
```

This problem of finding factorial of 5 can be broken down into a sub-problem of multiplying the factorial of 4 with 5.

```
5! = 5*4!
```

Or more generally,

```
n! = n*(n-1)!
```

Now we can continue this until we reach `0!` which is `1`.

The implementation of this is provided below.

Example: Recursive Function in R

```
# Recursive function to find factorial
recursive.factorial <- function(x) {
if (x == 0)    return (1)
else           return (x * recursive.factorial(x-1))
}
```

Here, we have a function which will call itself. Something like `recursive.factorial(x)` will turn into `x * recursive.factorial(x)` until `x` becomes equal to `0`.

When `x` becomes `0`, we return `1` since the factorial of `0` is `1`. This is the terminating condition and is very important.

Without this the recursion will not end and continue indefinitely (in theory). Here are some sample function calls to our function.

```
> recursive.factorial(0)
[1] 1
> recursive.factorial(5)
[1] 120
> recursive.factorial(7)
[1] 5040
```

The use of recursion, often, makes code shorter and looks clean.

However, it is sometimes hard to follow through the code logic. It might be hard to think of a problem in a recursive way.

Recursive functions are also memory intensive, since it can result into a lot of nested function calls. This must be kept in mind when using it for solving big problems.