Guía de Funciones en R

Larios Ponce Hector Manuel

22-07-2024

Introducción

Las funciones en R son fundamentales para estructurar y organizar el código. Permiten encapsular bloques de código que pueden ser reutilizados con diferentes parámetros. Esta guía cubre la creación, el uso y algunos conceptos avanzados relacionados con funciones en R.

Creación de Funciones

Sintaxis Básica

Para definir una función en R, se utiliza la siguiente sintaxis:

```
nombre_funcion <- function(argumento1, argumento2) {
  # Cuerpo de la función
  resultado <- argumento1 + argumento2
  return(resultado) # Retorno de la función
}
```

Donde:

- `nombre_funcion`: Identificador de la función.
- `argumento1`, `argumento2`: Parámetros de entrada de la función, puede o no requerirlos.
- Cuerpo de la función: Bloque de código que realiza las operaciones deseadas, opcional.
- `return()`: Sentencia para retornar un valor, opcional. Si no se utiliza, la función retornará el último valor calculado.

Ejemplo de Función

```
suma <- function(a, b) {
  resultado <- a + b
  return(resultado)
}
```

Llamada a una función

Para llamar a una función en R, se utiliza el identificador de la función seguido de los argumentos requeridos entre paréntesis:

```
# Llamar a la función
suma(3, 5)
```

```
## [1] 8
```

Funciones sin retorno explícito

En R, la última expresión evaluada en una función se devuelve automáticamente, por lo que no siempre es necesario usar `return()`.

```
sumar <- function(a, b) {
  a + b
}
# Llamar a la función
sumar(28, 15)
```

```
## [1] 43
```

Argumentos de Funciones

Argumentos con Valor por Defecto

En R, es posible asignar valores por defecto a los argumentos de una función. Esto permite que los argumentos sean opcionales y tomen un valor predeterminado si no se especifican al llamar a la función.

```
saludar <- function(nombre = "Mundo") {
  paste("Hola", nombre)
}
```

- Llamar a la función sin argumentos:

  ```
  saludar()
  ```

  ```
  ## [1] "Hola Mundo"
  ```

- Llamar a la función con un argumento:

  ```
  saludar("Juan")
  ```

  ```
  ## [1] "Hola Juan"
  ```

Argumentos con Nombre

Cuando una función tiene múltiples argumentos, es posible especificar los argumentos por nombre al llamar a la función. Esto permite cambiar el orden de los argumentos o especificar solo algunos de ellos.

```r
saludar_persona <- function(nombre, saludo = "Hola") {
  paste(saludo, nombre)
}
```

- Llamar a la función con argumentos por nombre:

```r
saludar_persona(nombre = "María", saludo = "Buenos días")
```

```
## [1] "Buenos días María"
```

- Llamar a la función con argumentos en diferente orden:

```r
saludar_persona(saludo = "Buenas noches", nombre = "Pedro")
```

```
## [1] "Buenas noches Pedro"
```

Argumentos de Longitud Variable

En R, es posible definir funciones con un número variable de argumentos utilizando `...` (Ellipsis). Esto permite que la función acepte cualquier cantidad de argumentos adicionales.

```r
# Definición de función con argumentos ellipsis
sumar_varios <- function(...) {
  numeros <- c(...)
  return(sum(numeros))
}
```

- Llamar a la función con argumentos variables:

```r
sumar_varios(1, 2, 3)
```

```
## [1] 6
```

```r
sumar_varios(10, 20, 30, 40, 50)
```

```
## [1] 150
```

```r
sumar_varios()
```

```
## [1] 0
```

Funciones anonimas

En R, las funciones anónimas (o funciones lambda) son funciones sin nombre que se pueden utilizar en expresiones o como argumentos de otras funciones. Se definen con la función `function()` y se pueden asignar a variables para su posterior uso.

A diferencia de las funciones regulares, las funciones anónimas no requieren un nombre, pueden tener una sintaxis más compacta y no pueden ser llamadas después de su definición.

```r
vector <- sapply(1:10, function(x) x^2)
```

En este ejemplo, se utiliza una función anónima para elevar al cuadrado cada elemento del vector del 1 al 10.

El vector resultante:

```r
print(vector)
```

```
##  [1]   1   4   9  16  25  36  49  64  81 100
```

Funciones Recursivas

Las funciones recursivas son aquellas que se llaman a sí mismas dentro de su definición. Son útiles para resolver problemas que se pueden dividir en casos más pequeños y similares al original.

Es importante tener en cuenta que las funciones recursivas deben tener dos componentes clave:

1. **Caso Base**: Condición que detiene la recursión y evita que la función se llame infinitamente.
2. **Llamada Recursiva**: Llamada a la función dentro de sí misma con argumentos modificados para acercarse al caso base.

```r
# Función recursiva para calcular el factorial de un número
factorial <- function(n) {
  if (n == 0) {
    return(1) # Caso base
  } else {
    return(n * factorial(n - 1)) # Llamada recursiva
  }
}
```

La llama a la función con un valor de 5 retornará:

```r
factorial(5)
```

```
## [1] 120
```