Funciones

En este documento explicaremos cómo definir y utilizar funciones en R. Explicaremos también algunas funciones útiles incorporadas con R, y algunas nociones básicas de programación funcional que podemos aplicar en nuestro código.

1. Definición y uso de funciones

Las funciones en R se crean con la palabra `function`, seguida de los parámetros que recibe la función y el código entre llaves. Esta función se asigna a una variable, de modo que, cuando queramos invocarla, usaremos esa variable.

```
suma <- function(n1, n2) {
    n1 + n2
}
```

A diferencia de otros lenguajes, en R no se utiliza la sentencia *return* para que la función *devuelva* un resultado. Simplemente ponemos el resultado en sí como una instrucción. Podemos después utilizar esta función a través de su nombre:

```
resultado <- suma(2, 3)   # 5
```

1.1. Nombres de parámetros y valores por defecto

Es importante el nombre que les damos a los parámetros, porque podemos alterar su orden cuando invocamos a la función. Por ejemplo, esta función eleva el parámetro *b* al exponente *e*:

```
potencia <- function(b, e) {
    b ^ e
}
```

Si invocamos a la función pasándole dos valores numéricos, el primero será la base *b* y el segundo el exponente *e*. Pero si indicamos cuál es cuál podemos invocar a la función pasando los parámetros en cualquier orden:

```
potencia(2, 3)     # 8
potencia(e=3, b=2) # 8
```

Por otra parte, podemos indicar un valor por defecto en los parámetros que queramos de modo que, si no se especifican, se toma ese valor por defecto:

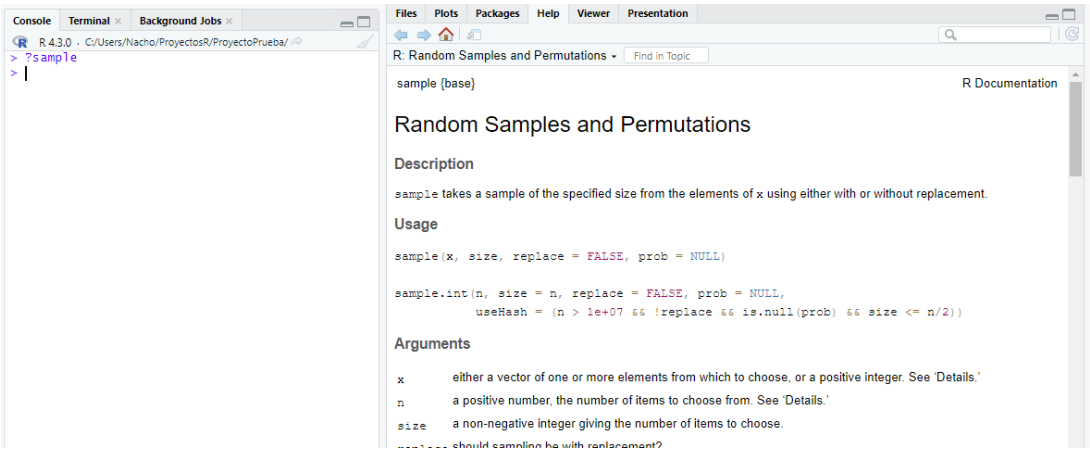
```
potencia <- function(b, e=2) {
    b ^ e
}

potencia(3, 3)    # 27
potencia(3)       # 9
```

2. Algunas funciones útiles

A lo largo de documentos anteriores ya hemos visto algunos ejemplos de funciones útiles que proporciona R, como la función `c` para crear vectores, o la función `length` para obtener el número de elementos de una colección.

Veamos ahora algunas funciones más que podemos emplear de las que ya vienen incorporadas en R. Para cualquier función que queramos utilizar podemos obtener ayuda en el editor *R Studio*, anteponiendo un interrogante al nombre de la función en la consola:

```
?sample
```



2.1. Generación de números aleatorios

Existen diferentes funciones que podemos emplear para generar números aleatorios:

- La función `sample` recibe como parámetros un vector de elementos y una cantidad entera, y devuelve un vector con esa cantidad de elementos tomada al azar del vector original. Opcionalmente admite un parámetro `replace=TRUE` para poder elegir repetidamente el mismo número

```
# Elegir 1 número al azar del vector (1, 2, 3)
elegido <- sample(c(1, 2, 3), 1)
# Elegir 2 números al azar (el resultado es otro vector)
elegidos <- sample(c(1, 2, 3), 2)
# Esto dará error porque no hay tantos números en el vector
elegidos2 <- sample(c(1, 2, 3), 4)
# Esto sí funcionará, y habrá números repetidos en la elección
elegidos3 <- sample(c(1, 2, 3), 4, replace=TRUE)
```

- La función `runif` permite generar números reales (con decimales). Le debemos indicar cuántos números generar y, opcionalmente, el valor mínimo (*min*) y máximo (*max*) del rango en que generarlos. Si estos límites no se especifican se toma el rango de 0 a 1.

```
# 5 números reales entre 0 y 10
numeros <- runif(5, min=0, max=10)
# 2 números reales entre 0 y 1
numeros2 <- runif(2)
```

2.2. Funciones matemáticas

Algunas funciones matemáticas que nos pueden resultar útiles para algunos cálculos son:

- `sqrt(x)` para calcular la raíz cuadrada
- `max(secuencia)` y `min(secuencia)` para obtener el máximo y mínimo valor de una secuencia
- `mean(secuencia)` para obtener la media de la secuencia de valores
- `round(x, n)` para redondear un número *x* con *n* decimales
- `floor(x)` para obtener el entero más cercano a x por debajo
- `ceiling(x)` para obtener el entero más cercano a x por encima

Aquí vemos algunos ejemplos:

```
resultado <- sqrt(81)           # 9
resultado <- max(c(1, 8, 3, 5))  # 8
resultado <- min(c(1, 8, 3, 5))  # 1
resultado <- mean(c(1, 8, 3, 5)) # 4.25
resultado <- round(3.1415926, 4) # 3.1416
resultado <- floor(3.1415926)    # 3
resultado <- ceiling(3.1415926)  # 4
```

3. Algunas nociones de programación funcional

La programación funcional es un paradigma de programación declarativo, donde indicamos qué características tiene el problema a resolver, en lugar de dar los pasos secuenciales para resolverlo. Algunas de sus características básicas son la composición de funciones (es decir, la posibilidad de pasar el resultado de una función como entrada a otra función), la inmutabilidad de los datos con que se trabaja y el uso de funciones de primer orden (es decir, funciones que se pasan como parámetro a otras funciones).

En lo que respecta a R, podemos hacer uso de algunas de estas características de la programación funcional, y aquí veremos algunas pinceladas al respecto. Por ejemplo, R dispone de una serie de funciones *apply* que aplican una función (que se pasa como parámetro) a los elementos de una colección:

La función `apply` aplica una función a una matriz de datos, por filas o columnas. El primer parámetro es la matriz, el segundo es un 1 para filas o 2 para columnas, y el tercer elemento es la función a aplicar:

```
matriz <- # ... Definimos la matriz
# Suma las filas de la matriz
sumas <- apply(matriz, 1, sum)
```

La función `lapply` es un caso particular de la anterior, que se aplica a listas o vectores unidimensionales (no a matrices)

```
datos <- c(1, 2, 3, 4)
# Raíces cuadradas de los elementos
raices <- lapply(datos, sqrt)
# Nota: obtendremos un vector de vectores, podemos convertirlo a simple con
resultado <- unlist(raices)
```

Podemos también definir la función en sí en la propia llamada:

```
datos <- c(1, 2, 3, 4)
# Cuadrados de los elementos
cuadrados <- unlist(lapply(datos, function(i) i ^ 2))
```

Existen otras variantes más, como `sapply` o `mapply`, que incluyen algunas particularidades que no veremos aquí.

> **Ejercicio 1:**
>
> Crea un programa `adivinar.r` que genere un vector de 3 números del 1 al 20 (inclusive), y lo deje ordenado. Después, le pedirá al usuario que diga 3 números y mostrará por pantalla cuántos números ha adivinado.

AYUDA: vídeo con solución del ejercicio