Elementos básicos del lenguaje

En este documento veremos los elementos básicos de que dispone el lenguaje R para construir nuestros primeros programas: tipos de datos básicos, salida de datos por pantalla, entrada de datos por parte del usuario, operadores elementales, etc.

1. Comentarios

Los comentarios en R comienzan con una almohadilla `#`, al estilo de otros lenguajes como Python. Son comentarios de una sola línea, que evitan ejecutar lo que vaya detrás de esa almohadilla.

```
# Comentario en R
print("Hola")
```

2. Tipos de datos básicos y variables

En R existen tres tipos de datos básicos:

- **Números**: pueden ser enteros (terminados en L: 3L, 242L), numéricos en general (25, 4.33) o incluso complejos (5 + 3i)
- **Textos**: también llamados tipo *character*, aunque R no dispone del tipo carácter (*char*) que sí existe en otros lenguajes. Así que para los textos podemos emplear indistintamente comillas dobles (`"Hola mundo"`) o simples (`'Hola mundo'`). También podemos emplear secuencias de escape habituales en otros lenguajes de programación, como el salto de línea `\n` o la tabulación `\t` (`"Hola\tmundo"`)
- **Booleanos**: pueden ser `TRUE` o `FALSE`, también abreviados como `T` o `F`, respectivamente.

2.1. Declaración e inicialización de variables

R es un lenguaje débilmente tipado, como Python, lo que significa que no tenemos que especificar de qué tipo es una variable, ni tenemos que declararla en un punto concreto. Simplemente se crean cuando les asignamos un valor. Para asignar un valor, a diferencia de muchos otros lenguajes, usaremos el operador `<-`, aunque también se admite el operador `=` habitual. También podemos asignar el mismo valor a varias variables en una sola instrucción, enlazando este mismo operador:

```
numero <- 3.5
n1 <- n2 <- n3 <- 10L
texto <- "Hola"
texto2 = "Adios"   # Usamos el operador =
```

El operador `<-` se puede poner en ambas direcciones. Lo que quede apuntado por la flecha es la variable donde se guardará el dato:

```
3.5 -> r
"Hola" -> texto
```

2.2. Conversión de datos

Para poder convertir entre distintos tipos simples podemos emplear las funciones `as.XXXX` correspondientes: `as.logical` (para pasar a booleano), `as.character` (para pasar a texto), `as.numeric` (para pasar a numérico general), `as.integer` (para pasar a entero)...

```
texto <- "3"
numero <- as.numeric(texto)
resultado <- numero * 2      # 6
```

Existen algunas conversiones implícitas. Por ejemplo, los datos falsos (*FALSE*) se codifican con un 0, y los verdaderos con un 1. Así, por ejemplo, si queremos ver cuántos elementos verdaderos hay en una secuencia de booleanos, podríamos sumar la secuencia y ver el resultado.

2.3. Detectar el tipo de dato de un elemento

La instrucción `class` permite obtener el tipo de dato de un elemento que le pasemos entre paréntesis:

```
class("hola") # "character"
```

3. Operadores

Veremos a continuación de forma resumida los principales operadores que ofrece el lenguaje R

3.1. Operadores aritméticos

Tenemos estos operadores aritméticos disponibles en R:

Operador	Descripción
`+`	Suma
`-`	Resta
`*`	Multiplicación
`/`	División
`%%`	Resto o módulo
`^`	Potencia
`%/%`	División entera

3.2. Operadores relacionales

A la hora de comparar dos elementos tenemos los típicos operadores de cualquier lenguaje: `==`, `!=`, `<`, `<=`, `>`, `>=`

3.3. Operadores lógicos

Los operadores lógicos que ofrece R para poder enlazar comprobaciones simples y formar otras más complejas son:

- `&`: Y lógica
- `|`: O lógica
- `!`: negación lógica

```
3 > 2 & 5 > 1
```

4. Entrada/Salida básica

Veamos ahora qué mecanismos ofrece R para sacar información por pantalla y pedirle datos al usuario

4.1. Salida por pantalla

Si estamos ejecutando R en una consola, para sacar el valor de un dato basta con escribir el dato y pulsar *Intro*. Alternativamente, desde código fuente podemos emplear la instrucción `print` para sacar el valor del dato que le pasemos como parámetro:

```
print ("Hola mundo")
```

Si queremos mostrar varios fragmentos enlazados, podemos emplear la instrucción `cat`. También es apropiada esta instrucción para incluir secuencias de escape, como `\n`:

```
cat("Hola", "mundo") # Hola mundo
cat("Buenos días\n")
```

Para formatear datos (especialmente valores numéricos) podemos emplear la instrucción `round` para redondear el número a los decimales adecuados, antes de imprimirlo. Esto funcionará siempre que el número de decimales real sea mayor que el que se pide mostrar (si es menor, se mostrará hasta la cantidad de decimales que se tenga). Se puede combinar con la función `format` y el parámetro *nsmall* para indicar cuántos decimales obtener en caso de que no haya suficientes:

```
numero <- 3.14159
numero2 <- 3.1
cat(round(numero, 2))                    # 3.14
cat(round(numero2, 2))                   # 3.1
cat(format(round(numero2, 2), nsmall=2)) # 3.10
```

4.2. Obtener la entrada del usuario

Existen varios mecanismos para leer datos de entrada. Uno bastante versátil es la instrucción `readLines`. Le pasaremos como primer parámetro el canal de entrada (en nuestro caso, para indicar teclado, usaremos `"stdin"`, y como segundo parámetro `n` cuántas líneas queremos leer (típicamente una). Obtendremos siempre la información en formato texto, y luego podemos emplear las funciones de conversión anteriores para convertir el dato al tipo adecuado:

```
cat("Dime tu nombre:\n")
nombre <- readLines("stdin", n=1)
cat("Dime tu edad:\n")
edad <- as.integer(readLines("stdin", n=1))
cat("Te llamas", nombre, "y tienes", edad, "años")
```

En algunos entornos de ejecución es posible que la instrucción `readLines` no funcione correctamente y no recoja datos de entrada. Alternativamente podemos emplear la instrucción `readline`, a la que le podemos indicar en el parámetro *prompt* el texto que queremos mostrar al usuario:

```
nombre <- readline(prompt="Dime tu nombre:\n")
edad <- as.integer(readline(prompt="Dime tu edad:\n"))
```

Ejercicio 1:

Crea un programa llamado `recorrido.r` que le pregunte al usuario una distancia en Km y una velocidad en Km/h y le diga cuánto tiempo (en horas) ha tardado en recorrer esa distancia a esa velocidad (incluyendo decimales).

AYUDA: vídeo con solución del ejercicio