2. Fundamentos

Índice

1. Comentarios

Los comentarios sirven para hacer anotaciones al código que no serán procesadas por el intérprete:

```
# Esto es un comentario
# Y esto otro
```

2. Variables

Esto son (asignaciones de) variables:

```
name <- 'Espagueti'
lastname <- "Volador"
age = 16
```

Podemos mostrar su valor:

```
name
print(age)
```

Y su tipo:

```
class(name)
```

Hacer asignaciones múltiples:

```
scorePlayer1 <- scorePlayer2 <- scorePlayer3 <- 5
```

Ejemplos de nombres válidos de variables:

```
fullname <- "Simone de Beauvoir"
full_name <- "Miyamoto Mushashi"
fullName <- "Ada Lovelace"
FULLNAME <- "Richard Feynman"
fullname2 <- "Clara Campoamor"
.fullname <- "Siddartha Gautama"
```

3. Tipos de datos

R presenta todos estos tipos básicos de datos:

- character/string:

  ```
  "Abracadabra pata de cabra"
  ```

- numeric:

  ```
  42
  3.14159
  ```

- integer:

  ```
  128L
  -273L
  ```

- complex:

  ```
  9 + 3i
  ```

- logical/boolean:

  ```
  TRUE
  FALSE
  ```

- NA (sin valor)

4. Operadores

Dependiendo del tipo de datos, este soportará unos operadores u otros:

- Los strings, concatenación:

```
fullname <- "Anonymous"
age <- 25

paste("Hola ", fullname, " tienes ", age)
```

- Los numéricos ofrecen operadores aritméticos:
  - Suma: `1 + 2`
  - Resta: `3 - 4`
  - Multiplicación: `6 * 7`
  - División: `10 / 3`
  - Potencia: `2 ^ 5`
  - Módulo: `10 %% 3`
  - División entera: `10 %/% 3`
- Los booleans, operadores lógicos:
  - Conjunción:

```
TRUE & FALSE
TRUE && FALSE
```

  - Disyunción:

```
TRUE | FALSE
TRUE || FALSE
```

  - Negación:

```
!TRUE
```

5. Coerciones

Es posible convertir variables de un tipo a otro si es necesario:

```
intValue <- 1L
numValue <- 2
strValue <- "34"

numValueFromInt <- as.numeric(intValue)
intValueFromNum <- as.integer(numValue)
numValueFromStr <- as.numeric(strValue)
```

6. Funciones matemáticas

El lenguaje ofrece un catálogo de funciones predefinidas:

```
max(5, 10, 15)
min(5, 10, 15)
sqrt(16)
abs(-4.7)
ceiling(1.4)
floor(1.4)
```

7. Manejo de cadenas

R permite strings multilínea:

```
myText <- "Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."
```

Donde R añade el carácter fin de línea (`\n`) automáticamente y para ver el texto interpretándolo en la salida, es necesario usar la función `cat`:

```
cat(myText)
myText
```

Para saber la longitud de un string, existe la función nchar:

```
nchar(myText)
```

Se puede comprobar si un substring está contenido en otro string con grepl:

```
grepl("tempor", myText)
grepl("totum", myText)
```

Permite escapar caracteres para que no sean interpretados:

```
boss <- "Me dijo que era un tanto \"impredecible\""
```

O usar caracteres especiales:

```
\\ \n \r \t \b
```

8. Manejo de fecha y hora

El primer paso es cargar los paquetes necesarios:

```
library(lubridate)
library(hms)
```

Algunas funciones útiles serían las siguientes:

```
today()
now()
ymd("1912 04 20")
mdy("1912 04 20")
hms(56,12,15)
ymd_hms("1912 04 20 23:19:59")
```

9. Estructuras de control de flujo

Para las estructuras de control de flujo se usan operadores lógicos (citados más arriba) y operadores de comparación:

```
== != < <= > >=
```

Condicionales

```
a <- 1
b <- 2
```

```
if (b > a) {
  "b es mayor que a"
}
```

Puede haber condicionales de dos o más ramas:

```
a <- 1
b <- 2
c <- 3

if (b > a) {
  "b es mayor que a"
} else {
  "a es mayor que b"
}

if (a > c) {
  "a es mayor que c"
} else if (a > b) {
  "a es mayor que b"
} else {
  "a es el más pequeño de todos"
}
```

Bucles

Un bucle debe tener tres elementos:

- Inicialización previa
- Condición de parada
- Cambios en cada iteración

While

```
i <- 0

while (i < 5) {
  print(i)
  i <- i + 1
}
```

For

```
for (i in 1:5) {
  print(i)
}

fruits <- list("cereza", "pomelo", "coco")

for (f in fruits) {
  print(f)
}
```

Ambos tipos de bucles permiten el uso de `next` y `break` para romper el control de flujo:

```
i <- 1
while (i < 5) {
  print(i)
  i <- i + 1

  if (i == 3) {
    break
  }
}

fruits <- list("cereza", "pomelo", "coco")

for (f in fruits) {
  if (f == "pomelo") {
    next
  }

  print(f)
}
```

Referencias

Comentarios
Variables
Tipos de datos
Cadenas
Números
Lógicos
Operadores
Funciones matemáticas
Manejo de fecha y hora
Condicionales
Bucles While
Bucles For