

Bloque 1: Fundamentos y Tipos de Datos

R es un lenguaje de programación orientado al análisis estadístico y la visualización de datos. En este primer bloque aprenderemos las bases: variables, tipos de datos y las operaciones más elementales.

1.1 Variables y tipos básicos

En R no es necesario declarar el tipo de una variable. La asignación se realiza con el operador `<-` (se lee 'asignar').

```
# Tipos de datos basicos
numero <- 42          # numerico (numeric)
decimal <- 3.14       # decimal (double)
texto <- "Hola R"    # cadena de texto (character)
logico <- TRUE        # logico (logical): TRUE o FALSE

# Ver el tipo de una variable
class(numero)        # "numeric"
class(texto)         # "character"
typeof(logico)       # "logical"

# Ver todos los objetos en memoria
ls()
```

Consejo

Usa `<-` para asignar variables (es el estándar en R). El operador `=` también funciona, pero se reserva para los argumentos de funciones.

1.2 Operaciones aritméticas

```
# Operadores aritmeticos
5 + 3    # suma      -> 8
5 - 3    # resta     -> 2
5 * 3    # producto  -> 15
5 / 3    # division  -> 1.666...
5 %% 3   # modulo (resto) -> 2
5 %/% 3  # division entera -> 1
5 ^ 2    # potencia  -> 25
```

Ejercicio 1: Variables y tipos

Crea tres variables: tu nombre, tu edad y si eres estudiante. Muestra el tipo de cada una.

```
nombre <- "Ana"
edad <- 22
estudiante <- TRUE

class(nombre) # "character"
class(edad) # "numeric"
class(estudiante) # "logical"

cat("Hola,", nombre, "tienes", edad, "años.\n")
```

Explicación: cat() concatena e imprime con formato. \n es un salto de línea. class() nos dice el tipo de cada variable.

Ejercicio 2: Redondear y truncar

Trunca el número pi a exactamente 2 decimales SIN usar round().

```
p <- 3.141516

# Multiplicamos por 100, truncamos la parte decimal,
# y volvemos a dividir por 100
trunc(p * 100) / 100 # -> 3.14

# Comparación: round() redondea, trunc() elimina
round(3.145, 2) # -> 3.15 (redondea hacia arriba)
trunc(3.145*100)/100 # -> 3.14 (corta sin redondear)
```

Explicación: trunc() elimina la parte decimal hacia cero. Al multiplicar por 100 y dividir después desplazamos el punto decimal para controlar la precisión.

Ejercicio 3: Divisibles entre 6 y 8

Cuántos números entre 1 y 1.000.000 son divisibles a la vez por 6 y por 8?

```
numeros <- 1:1000000

# %% es el operador modulo (resto de la división)
condicion <- numeros %% 6 == 0 & numeros %% 8 == 0

sum(condicion) # -> 41667

# Alternativa: divisibles por el mcm(6,8) = 24
sum(numeros %% 24 == 0) # mismo resultado
```

Explicación: R opera sobre vectores enteros de forma vectorizada, sin necesidad de bucles. sum() cuenta los TRUE (que valen 1 en R).

Ejercicio 4: Operaciones con pi

Calcula el area y el perimetro de una circunferencia de radio 7. Usa la constante pi de R.

```
radio <- 7

area      <- pi * radio^2
perimetro <- 2 * pi * radio

cat("Area:      ", round(area, 4), "\n")
cat("Perimetro:", round(perimetro, 4), "\n")

# R tiene pi como constante incorporada
pi # -> 3.141593
```

Explicación: R incluye la constante pi precargada. round(x, n) redondea x a n decimales.

Bloque 2: Vectores y Operaciones Vectoriales

El vector es la estructura de datos más fundamental en R. Todo en R es, en última instancia, un vector. Entender los vectores es dominar la mitad del lenguaje.

2.1 Creación de vectores

```
# Formas de crear vectores
v1 <- c(10, 20, 30, 40, 50)      # funcion c()
v2 <- 1:10                       # secuencia de enteros
v3 <- seq(0, 1, by = 0.25)       # secuencia con paso
v4 <- rep(5, times = 4)          # repeticion: 5 5 5 5

# Acceso por indice (los indices en R empiezan en 1)
v1[1]                           # primer elemento -> 10
v1[c(2, 4)]                     # elementos 2 y 4 -> 20 40
v1[-1]                          # todos menos el primero -> 20 30 40 50
```

2.2 Operaciones vectoriales

```
# R opera elemento a elemento (vectorizado)
x <- c(1, 2, 3, 4, 5)
x * 2      # -> 2  4  6  8 10
x > 3      # -> FALSE FALSE FALSE TRUE TRUE
sum(x)     # -> 15
mean(x)    # -> 3
sd(x)      # desviacion tipica
var(x)     # varianza
cumsum(x)  # suma acumulada: 1 3 6 10 15

# Filtrado logico
x[x > 3]    # -> 4 5
which(x > 3) # indices donde se cumple: 4 5
```

Ejercicio 5: Análisis de compras

Tienes 1000 compras aleatorias con media 5000 y desviación 1000. Responde las preguntas.

```
set.seed(42)
compras <- rnorm(n = 1000, mean = 5000, sd = 1000)

# Suma de las posiciones 435 y 678
sum(compras[c(435, 678)])

# Cuantas compras superan 8000 euros
sum(compras > 8000)

# Importe maximo y minimo
max(compras)
min(compras)

# Porcentaje de compras entre 4000 y 6000
sum(compras >= 4000 & compras <= 6000) / length(compras)
```

Explicación: rnorm() genera números aleatorios con distribución normal. set.seed() fija la semilla para reproducibilidad.

Ejercicio 6: Estadísticas de un vector

Crea un vector con las notas de 10 alumnos (del 0 al 10) y calcula: media, mediana, nota máxima, mínima, cuantos aprobaron (≥ 5) y el porcentaje de aprobados.

```
set.seed(99)
notas <- round(runif(10, 0, 10), 1)
notas # muestra el vector

mean(notas)      # media
median(notas)    # mediana
max(notas)       # maxima
min(notas)       # minima

aprobados <- sum(notas >= 5)
aprobados

porc_aprobados <- aprobados / length(notas) * 100
cat(porc_aprobados, "% de aprobados\n")
```

Explicación: runif() genera números uniformemente distribuidos entre min y max. Combinado con round() obtenemos notas con un decimal.