Módulo 2: Tipos de Datos y Estructuras Básicas en R

12 de June de 2025

1. Tipos de Datos:

A) Numéricos:

Los números en R pueden ser enteros o decimales. Por defecto, R maneja numeric como double (decimales). Para un entero explícito, se usa L.

- **Ejemplos de números**
  - a <- 3.14 # numeric (double)
  - b <- 2L # usando L se define un entero (integer)

```
## [1] 3.14
## [1] 2
```

1. Tipos de Datos:

A) Numéricos:

- **Verificando tipos**
  - is.numeric(a)
  - is.integer(b)

```
## [1] TRUE
## [1] TRUE
```

1. Tipos de Datos:

A) Numéricos:

- **Operaciones básicas siempre devuelven "numeric" (double)**
  - 5 / 2
  - 5L / 2L
  - 5L %/% 2L # División entera

```
## [1] 2.5
## [1] 2.5
## [1] 2
```

1. Tipos de Datos:

B) Caracteres:

Cadenas de texto encerradas en comillas simples o dobles.

- **Ejemplos de cadenas de texto**
  - nombre <- “Juan”
  - saludo <- ‘Hola, mundo’

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

1. Tipos de Datos:

B) Caracteres:

- **Verificando clase y longitud**
  - class(nombre)
  - nchar(saludo)

```
## [1] "character"
## [1] 11
```

1. Tipos de Datos:

B) Caracteres:

- **Funciones útiles para caracteres**
  - paste(“Hola”, “RStudio”, sep = ” “)
  - paste0(“A”, 1:3)

```
## [1] "Hola RStudio"
## [1] "A1" "A2" "A3"
```

1. Tipos de Datos:

B) Caracteres:

- **cadena_larga <- “La noche estrellada de Van Gogh”**
  - substr(cadena_larga, 4, 9)
  - substring(cadena_larga, 15, 23)

```
## [1] "noche "
## [1] "llada de "
```

1. Tipos de Datos:

B) Caracteres

- **texto_variado <- c(“azul”, “cielo”, “estrellas”, “noche”, “sol”)**
  - grep(“noche”, texto_variado)
  - grepl(“estrellas”, texto_variado)

```
## [1] 4
## [1] FALSE FALSE  TRUE FALSE FALSE
```

1. Tipos de Datos:

C) Lógicos

Solo toman valores TRUE o FALSE (pueden abreviarse T y F si no hay ambigüedad).

- **Ejemplos de valores lógicos**
  - x <- 5
  - y <- 3
  - z <- x > y
  - is.logical(z)

```
## [1] 5
## [1] 3
## [1] TRUE
## [1] TRUE
```

1. Tipos de Datos:

C) Lógicos

- **Operaciones lógicas**
  - (x $>$ 4) & (y $<$ 10)
  - (x $>$ 4) && (y $<$ 10)
  - (x $<$ 4) | (y $<$ 10)
  - (x $<$ 4) || (y $<$ 10)
  - !z

```
## [1] TRUE
## [1] TRUE
## [1] TRUE
## [1] TRUE
## [1] FALSE
```

1. Tipos de Datos:

D) Factores

Representan variables categóricas con niveles definidos.

Internamente se almacenan como enteros con etiquetas.

- **Creación de un factor**
  - sexo <- factor(c("M", "F", "F", "M"), levels = c("M", "F"))
  - print(sexo)
  - levels(sexo)
  - table(sexo)

```
## [1] M F F M
## Levels: M F
## [1] "M" "F"
## sexo
## M F
## 2 2
```

1. Tipos de Datos:

D) Factores

- **Funciones útiles**
  - colores_factor <- as.factor(c("rojo", "azul", "verde", "rojo"))
  - print(colores_factor)

```
## [1] rojo  azul  verde rojo
## Levels: azul rojo verde
```

1. Tipos de Datos:

D) Factores

- **Funciones útiles**
  - colores_reordenados <- factor(colores_factor, levels = c("verde", "azul", "rojo"))
  - levels(colores_reordenados)

```
## [1] "verde" "azul"  "rojo"
```

1. Tipos de Datos:

D) Factores

- **Funciones útiles**
  - satisfaccion <- ordered(c("baja", "media", "alta"), levels = c("baja", "media", "alta"))
  - print(satisfaccion)

```
## [1] baja  media alta
## Levels: baja < media < alta
```

2. Vectores:

Los vectores son colecciones de elementos del mismo tipo.

- **Creación con c()**
  - v1 <- c(1, 2, 3, 4)
  - v2 <- c("a", "b", "c")
  - v3 <- c(TRUE, FALSE, TRUE)
  - print(v1)
  - print(v2)
  - print(v3)

```
## [1] 1 2 3 4
## [1] "a" "b" "c"
## [1]  TRUE FALSE  TRUE
```

2. Vectores:

- **Secuencias con seq()**
  - seq(1, 10, by = 2)
  - seq(5, 15, length.out = 6)

```
## [1] 1 3 5 7 9
## [1]  5  7  9 11 13 15
```

2. Vectores:

- **Repetición con rep()**
  - rep(1:3, times = 2)
  - rep(c("X", "Y"), each = 3)

```
## [1] 1 2 3 1 2 3
## [1] "X" "X" "X" "Y" "Y" "Y"
```

2. Vectores:

Indexación y Operaciones con Vectores

- **Indexación**
  - v <- c(10, 20, 30, 40, 50)
  - v[2] # Por posición
  - v[1:3]
  - v[-1] # Índices negativos (excluyen)
  - v[v > 25] # Por índice lógico (filtrado)

```
## [1] 20
## [1] 10 20 30
## [1] 20 30 40 50
## [1] 30 40 50
```

2. Vectores:

- **Operaciones elementales (vectorización)**
  - v_a <- c(1, 2, 3)
  - v_b <- c(4, 5, 6)
  - v_a + v_b
  - v_a == 2

```
## [1] 5 7 9
## [1] FALSE  TRUE FALSE
```

2. Vectores:

- **Funciones vectorizadas**
  - numeros <- c(10, 5, 20, 15, 30)
  - sum(numeros)
  - mean(numeros)
  - sd(numeros)

```
## [1] 80
## [1] 16
## [1] 9.617692
```

2. Vectores:

- **Funciones vectorizadas**
  - min(numeros)
  - max(numeros)
  - length(numeros)
  - sort(numeros)
  - order(numeros)

```
## [1] 5
## [1] 30
## [1] 5
## [1]  5 10 15 20 30
## [1] 2 1 4 3 5
```

3. Matrices:

Las matrices son colecciones bidimensionales de elementos del mismo tipo.

- **Crear con matrix()**
  - m <- matrix(data = 1:6, nrow = 2, ncol = 3, byrow = TRUE)
  - print(m)

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

3. Matrices:

- **Asignar nombres de filas y columnas**
  - rownames(m) <- c("Fila1", "Fila2")
  - colnames(m) <- c("ColA", "ColB", "ColC")
  - print(m)

```
##       ColA ColB ColC
## Fila1    1    2    3
## Fila2    4    5    6
```

3. Matrices:

- **Indexación de matrices**
  - m[“Fila2”, “ColB”]
  - m[1,]
  - m[, 2]
  - m[1, 3]

```
## [1] 5
## ColA ColB ColC
##    1    2    3
## Fila1 Fila2
##     2     5
## [1] 3
```

3. Matrices:

Operaciones con Matrices

- **Operaciones elemento a elemento**
  - m * 2
  - m * m

```
##       ColA ColB ColC
## Fila1    2    4    6
## Fila2    8   10   12
##       ColA ColB ColC
## Fila1    1    4    9
## Fila2   16   25   36
```

3. Matrices:

- **Producto matricial**
  - n <- matrix(1:6, nrow = 3, ncol = 2)
  - m %*% n

```
##       [,1] [,2]
## Fila1   14   32
## Fila2   32   77
```

3. Matrices:

- **Transposición**
  - t(m)

```
##      Fila1 Fila2
## ColA     1     4
## ColB     2     5
## ColC     3     6
```

3. Matrices:

- **Determinante e Inversa (para matrices cuadradas)**
  - m_cuadrada <- matrix(c(4, 7, 2, 6), nrow = 2, ncol = 2, byrow = TRUE)
  - det(m_cuadrada)
  - solve(m_cuadrada)

```
##      [,1] [,2]
## [1,]    4    7
## [2,]    2    6
## [1] 10
##      [,1] [,2]
## [1,]  0.6 -0.7
## [2,] -0.2  0.4
```

4. Listas:

Las listas pueden contener cualquier tipo de objeto: vectores, matrices, data frames, funciones, otras listas.

- **Definición con list()**
  - mi_lista <- list(vec = c(1, 2, 3),mat = matrix(4:9, nrow = 2),texto = "Hola",otro = list(sub1 = TRUE, sub2 = c("a", "b")))
  - print(mi_lista)

4. Listas:

```
## $vec
## [1] 1 2 3
##
## $mat
##      [,1] [,2] [,3]
## [1,]    4    6    8
## [2,]    5    7    9
##
## $texto
## [1] "Hola"
##
## $otro
## $otro$sub1
## [1] TRUE
##
```

4. Listas:

- **Acceso a elementos**
  - mi_lista[[1]] # Por posición
  - mi_lista[[4]][[“sub2”]]
  - mi_lista$texto # Por nombre
  - mi_lista*otro*sub1

```
## [1] 1 2 3
## [1] "a" "b"
## [1] "Hola"
## [1] TRUE
```

4. Listas:

- **Modificar y agregar elementos**
  - mi_lista$vec <- c(10, 20, 30)
  - mi_lista[[“nuevo_elem”]] <- rnorm(5)
- **Funciones útiles**
  - length(mi_lista)
  - names(mi_lista)

```
## [1] 5
## [1] "vec"       "mat"       "texto"     "otro"
```

5. Data Frames:

Los Data Frames son estructuras tabulares donde cada columna puede ser de un tipo diferente.

- **Creación con data.frame()**
  - df <- data.frame(id = 1:4,nombre = c("Ana", "Luis", "María", "Pedro"),edad = c(23, 35, 28, 42),sexo = factor(c("F", "M", "F", "M")))
  - print(df)

```
##   id nombre edad sexo
## 1  1    Ana   23    F
## 2  2   Luis   35    M
## 3  3  María   28    F
## 4  4  Pedro   42    M
```

5. Data Frames:

- **Inspección rápida**
  - dim(df)
  - nrow(df)
  - ncol(df)
  - str(df)
  - summary(df)

```
## [1] 4 4
## [1] 4
## [1] 4
## 'data.frame':    4 obs. of  4 variables:
##  $ id    : int  1 2 3 4
##  $ nombre: chr  "Ana" "Luis" "María" "Pedro"
##  $ edad  : num  23 35 28 42
##  $ sexo  : Factor w/ 2 levels "F","M": 1 2 1 2
##        id          nombre                edad       sexo
```

5. Data Frames:

Selección y Manipulación de Data Frames

- **Selección de columnas**
  - df$edad
  - df[, 2]
  - df[, “sexo”]
  - df[, c(“nombre”, “edad”)]

```
## [1] 23 35 28 42
## [1] "Ana"   "Luis"  "María" "Pedro"
## [1] F M F M
## Levels: F M
##   nombre edad
## 1    Ana   23
## 2   Luis   35
## 3  María   28
## 4  Pedro   42
```

5. Data Frames:

- **Selección de filas**
  - df[1:2,]
  - df[df$edad $>$ 30,] # Condicional

```
##   id nombre edad sexo
## 1  1    Ana   23    F
## 2  2   Luis   35    M
##   id nombre edad sexo
## 2  2   Luis   35    M
## 4  4  Pedro   42    M
```

5. Data Frames:

- **Agregar y eliminar columnas**
  - df$salario <- c(50000, 65000, 48000, 72000)
  - print(df)
  - df$edad <- NULL # Elimina la columna
  - print(df)

```
##   id nombre edad sexo salario
## 1  1    Ana   23    F   50000
## 2  2   Luis   35    M   65000
## 3  3  María   28    F   48000
## 4  4  Pedro   42    M   72000
##   id nombre sexo salario
## 1  1    Ana    F   50000
## 2  2   Luis    M   65000
## 3  3  María    F   48000
## 4  4  Pedro    M   72000
```

5. Data Frames:

- **Agregar y eliminar filas**
  - nueva_fila <- data.frame(id = 5, nombre = "Carlos", sexo = factor("M"), salario = 58000)
  - df <- rbind(df, nueva_fila)
  - print(df)
  - df <- df[-3,] # Elimina la fila 3
  - print(df)

```
##   id nombre sexo salario
## 1  1    Ana    F   50000
## 2  2   Luis    M   65000
## 3  3  María    F   48000
## 4  4  Pedro    M   72000
## 5  5 Carlos    M   58000
##   id nombre sexo salario
## 1  1    Ana    F   50000
```

6. Coercion de Tipos:

La coerción es la conversión de un tipo de dato a otro, explícita o implícita.

- **Conversión explícita con funciones as.**
  - x_char <- c(“1”, “2”, “3”)
  - x_num <- as.numeric(x_char)
  - print(x_num)
  - y_fac <- as.factor(x_num)
  - print(y_fac)
  - z_char <- as.character(y_fac)
  - print(z_char)

6. Coercion de Tipos:

- **Conversión implícita**
- **R coerciona al tipo más general al combinar elementos**
  - mezcla <- c(1, “a”, TRUE)
  - print(mezcla) # Resultado: “1”, “a”, “TRUE” (character)

6. Coercion de Tipos:

- **Comprobar tipos**
  - typeof(x_num)
  - is.numeric(x_num)
  - is.character(x_char)
  - is.factor(y_fac)