Clase 2 (Beta)

Alexis Zúñiga & Ernesto Barrios

Capítulo 4: Uso Basico de R

Empezemos a familiarizarnos con el entorno de R, especificamente con su sintaxis.

Declaración de variables

R tiene un operador especial para asignar valores a variables, diferentes a otros lenguajes de programación. Además, las variables son flexibles y pueden pueden pasar de contener un tipo de dato a otro sin problema. Por los mismo no es necesario especificar el tipo de dato como se hace en C o en Java. El operador de asignación es: `<-` o `->`.

También se puede asignar variables con = (al igual que en Python) pero, por convención, = se reserva para operaciones dentro de funciones o paréntesis.

Vamos a asignar algunos valores a una serie de variables:

```
numero1 <- 10
0.125 -> numero2
entero <- 20
texto <- "ejemplo"
texto2 <- 'también se puede con comillas simples'
boolenao = TRUE
booleano1 <- T
boolenao2 = FALSE
booleano22 <- F

typeof(numero1)
```

```
## [1] "double"
```

```
typeof(entero)
```

```
## [1] "double"
```

Como describimos, no se necesita especificar qué tipo de dato queremos en cada varible ya que esto puede cambiar más adelante. Hablando de tipos de datos, vamos a ver cuáles son las opciones que maneja R:

Tipo de datos

En R, los tipos de datos más comunes son:

1. **Numérico (numeric)**: Se utilizan para representar números reales, ya sea enteros o decimales.
2. **Entero (integer)**: Utilizado para representar números enteros.
3. **Carácter (character)**: Utilizado para representar texto. Las cadenas de texto se escriben entre comillas simples o dobles.
4. **Lógico (logical)**: Representa valores booleanos que pueden ser `TRUE` o `FALSE`.

5. **Complejo (complex)**: Utilizado para representar números complejos con una parte real y una imaginaria.
6. **Fecha y hora (date y datetime)**: Existen clases específicas para manejar fechas y horas.
7. **Factores (factor)**: Se utilizan para representar variables categóricas.
8. **Listas (list)**: Estructura de datos que puede contener elementos de diferentes tipos.
9. **Vectores (vector)**: Colecciones ordenadas de elementos del mismo tipo, como vectores numéricos, de caracteres, lógicos, etc.

Estos son los tipos de datos más utilizados en R, pero existen otros tipos más especializados y estructuras de datos complejas como matrices, data frames, arrays, entre otros.

```r
# Ejemplo de diferentes tipos de datos en R

# Numérico
numero_real <- pi
print(numero_real)
```

```
## [1] 3.141593
```

```r
typeof(numero_real)
```

```
## [1] "double"
```

```r
# Entero
(numero_entero <- 42L)
```

```
## [1] 42
```

```r
typeof(numero_entero)
```

```
## [1] "integer"
```

```r
# Carácter
texto <- "Hola, mundo!"
print(texto)
```

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

```r
typeof(texto)
```

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

```r
# Lógico
(es_verdadero <- TRUE)
```

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

```r
typeof(es_verdadero)
```

```
## [1] "logical"
```

```r
# Complejo
numero_complejo <- 2 + 3i
print(numero_complejo)
```

```
## [1] 2+3i
```

```r
typeof(numero_complejo)
```

```
## [1] "complex"
```

```r
# Fecha y hora
fecha <- as.Date("2022-12-31")
print(fecha)
```

```
## [1] "2022-12-31"
```

```r
typeof(fecha)
```

```
## [1] "double"
```

```r
# Factores
factor_variable <- factor(c("A", "B", "A", "C", "B"))
print(factor_variable)
```

```
## [1] A B A C B
## Levels: A B C
```

```r
typeof(factor_variable)
```

```
## [1] "integer"
```

```r
# Listas
lista <- list(nombre = "Juan", edad = 30, casado = TRUE)
print(lista)
```

```
## $nombre
## [1] "Juan"
##
## $edad
## [1] 30
##
## $casado
## [1] TRUE
```

```r
typeof(lista)
```

```
## [1] "list"
```

```r
# Vectores
(vector_numerico <- c(1, 2, 3, 4, 5))
```

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

```r
typeof(vector_numerico )
```

```
## [1] "double"
```

En R , cuando utilizas la función `typeof()` para conocer el tipo de datos de una variable que almacena un número, es común que obtengas "double" como respuesta. Esto se debe a que R trata la mayoría de los números como valores de tipo double, que son números de punto flotante de doble precisión. Por lo tanto, incluso si introduces un número entero, R lo interpretará como un double. Es una característica de cómo R maneja los números internamente.

Operadores

La mayor parte de los operadores de R van a ser similares a los de otros sistemas operativos, veamos algunos ejemplos:

Operadores aritméticos

```
#Suma(+)
(2+2)
```

```
## [1] 4
```

```
#Resta(-)
(3-2)
```

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

```
#Multiplicación(*)
(2*4)
```

```
## [1] 8
```

```
#División(/)
(8/2)
```

```
## [1] 4
```

```
#Potencia(**)
(5**2)
```

```
## [1] 25
```

```
#Raíz
(25**(1/2))
```

```
## [1] 5
```

```
#La raíz tambien puede obtenerse por la funicón sqtr()
(sqrt(36))
```

```
## [1] 6
```

```
#Módulo(residuo de una división)(%%)
(5%%2)
```

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

Tambien se puede hacer operación entre variables:

```
x <- 5
y <- 2
(suma <- x+y)
```

```
## [1] 7
```

```
#otra forma:
(x+y)
```

```
## [1] 7
```

```
(resta<-x-y)
```

```
## [1] 3
```

```
(multiplicacion <- x*y)
```

```
## [1] 10
```

```
division <- x/y
print(division)
```

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

```
(potencia <- x**y)
```

```
## [1] 25
```

```
#otra manera de usar la potencia es con ^
(potencia2 <- x^y)
```

```
## [1] 25
```

```
(modulo <- x%%y)
```

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

Operadores de comparación:

```
a <- 5
b <- 10
(igualdad <- x == y)
```

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

```
(desigualdad <- x != y)
```

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

```
(menor_que <- x < y)
```

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

```
(mayor_que <- x > y)
```

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

```
(menor_o_igual <- x <= y)
```

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

```
(mayor_o_igual <- x >= y)
```

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

Operadores lógicos

```
p <- TRUE
q <- FALSE
(and_logico <- p & q)
```

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

```
(or_logico <- p | q)
```

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

```
(not_logico <- !p)
```

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

Concatenación

En R se usa la funicón `paste()` para concatenar cadenas de texto.

```
nombre <- "Juan"
apellido <- "Pérez"
nombre_completo <- paste(nombre, apellido)
print(nombre_completo)
```

```
## [1] "Juan Pérez"
```

Operador de existencia

Para verificar la existencia de un valor en un vector, usamos la función

```
a <- c(1, 2, 3, 4, 5)
b <- 3
(resultado <- b %in% a)
```

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

Algunas otras operaciones comunes:

```
 # log
    log(1)
```

```
## [1] 0
```

```
    log(10, base = 10)
```

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

```
    log(0) #Ojo
```

```
## [1] -Inf
```

```
  # exp
    exp(1)
```

```
## [1] 2.718282
```

```
    exp(log(2))
```

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

Capítulo 5: Vectores

"Los vectores en En R son objetos de una sola dimensión que pueden contener datos númericos, cadenas de caracter, o datos logicos, entre otros. Esencialmente son uno de los elementos básicos en la estructura de los datos en R . Pueden contener solo elementos de un solo tipo, aunque su tamaño podría ser ilimitado"(Viva el software libre, s.f.). Veamos ahora ejemplo de manipulación de vectores.

Creación de vectores

Los vectores se crean usando la función combine: `c()`

```
vector_numerico <- c(1, 3, 5, 7)
vector_texto <- c("a", "b", "c", "d")
vector_logico <- c(TRUE, FALSE, TRUE)
vector_numerico_ordenado <-c(1:12)
print(vector_numerico_ordenado)
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

La moda de un vector puede obtenerse con la funcion `mode()`

```
(x <- c())
```

```
## NULL
```

```
(a <- c(1, 2, 3))
```

```
## [1] 1 2 3
```

```
mode(a)
```

```
## [1] "numeric"
```

```
(y <- c("hola", 2, 3))
```

```
## [1] "hola" "2"    "3"
```

```
mode(y)
```

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

```
(a <- c(1, 2, 3, 4, 5 + 3i))
```

```
## [1] 1+0i 2+0i 3+0i 4+0i 5+3i
```

```
mode(a)
```

```
## [1] "complex"
```

Notar que los elementos son forzados a ser del mismo tipo.

Ademas del tipo de datos, otra caracteristica importante de un vector es su longitud, que puede obtenerse con la funcion `length()`.

```
length(a)
```

```
## [1] 5
```

Que pasa con la dimension?

```
dim(a)
```

```
## NULL
```

Operaciones de vectores

Las operaciones entre vectores van a funcionar igual que las operaciones normales, sin embargo se tienen que cumplir algunas condiciones:

```r
y <- c(1,2,3,4,5,6)
print(y)
```

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

```r
# Si los vectores no son de la misma longitud:
  y + c(-1, 1, -1, 1,-1, 1, -1, 1)
```

```
## Warning in y + c(-1, 1, -1, 1, -1, 1, -1, 1): longer object length is not a
## multiple of shorter object length
```

```
## [1] 0 3 2 5 4 7 0 3
```

```r
# Los vectores mas cortos se repiten hasta tener la longitud del vector mas largo

  y * c(1, 2, 3)
```

```
## [1]  1  4  9  4 10 18
```

Notese que3 el vector atómico es la estructura fundamental de R . Eso implica que si no se especifica como guardar una estructura de datos, R la gaurdara como vector.

```r
(vec <- 1:5)
```

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

Por ultimo, cada elemento del vector puede tener un nombre y se le asigna de la siguiente manera:

```r
(vec2 <- c(elemtento1=47484, elemento2= "akfaj"))
```

```
## elemtento1  elemento2
##    "47484"    "akfaj"
```