Tipos y estructuras de datos

J. Eduardo Gamboa U.

2024-10-06

Tipos de datos estructurados

Los datos estructurados pueden ser dispuestos en un formato tabla (filas y columnas). Son los tipos de datos más sencillos, pero no los más comunes

Integer

Permite representar números enteros, ¿qué sucede al ejecutar estos códigos?

```
x = 4L
typeof(x)
```

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

```
is.integer(x)
```

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

```
y = 4
typeof(y)
```

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

```
is.integer(y)
```

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

Double

Permite representar números reales (permite decimales), ¿qué sucede al ejecutar estos códigos?

```
x = 36
typeof(x)
```

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

```
is.integer(x)
```

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

```
is.double(x)
```

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

```
y = 40.3
typeof(y)
```

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

```
is.integer(y)
```

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

```
is.double(y)
```

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

Complex

Permite representar números complejos, ¿qué sucede al ejecutar estos códigos?

```
z = 1 + 2i
typeof(z)
```

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

```
is.integer(z)
```

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

```
is.double(z)
```

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

```
is.complex(z)
```

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

```
t = 1i
typeof(t)
```

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

```
is.integer(t)
```

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

```
is.double(t)
```

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

```
is.complex(t)
```

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

```
w = 10 + 0i
typeof(w)
```

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

```
is.integer(w)
```

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

```
is.double(w)
```

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

```
is.complex(w)
```

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

```
v = 10
typeof(v)
```

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

```
is.integer(v)
```

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

```
is.double(v)
```

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

```
is.complex(v)
```

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

Logical

Permite representar datos lógicos o booleanos, los cuales admiten los valores verdadero (T o TRUE) o falso (F o FALSE), ¿qué sucede al ejecutar estos códigos?

```
a = TRUE
typeof(a)
```

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

```
is.integer(a)
```

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

```
is.double(a)
```

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

```
is.complex(a)
```

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

```
is.logical(a)
```

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

```
3 == 5.2
```

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

```
4.6 <= 8.1
```

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

```
12 > 3.5
```

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

```
5 != 8
```

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

```
b = (4 == 5)
```

Character

Permite representar datos como cadenas no numéricas, es decir letras, símbolos y/o valores alfanuméricos. Estos datos ya se podrían considerar como no estructurados, ¿qué sucede al ejecutar estos códigos?

```r
p = 'Universidad Agraria'
typeof(p)
```

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

```r
is.integer(p)
```

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

```r
is.double(p)
```

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

```r
is.complex(p)
```

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

```r
is.logical(p)
```

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

```r
is.character(p)
```

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

```r
q = '70096321'

s = 'R is a free software environment for statistical computing and graphics. To download R, please choo
```

Colecciones de datos

Vector atómico

Se trata de una colección de n datos del mismo tipo (Integer, Double, Logical, Character, Complex). De ser necesario, aplica coerción implícita. Por ejemplo:

```r
x1 = c(3,4,5,12,3.4)
typeof(x1)
```

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

```r
str(x1)
```

```
##  num [1:5] 3 4 5 12 3.4
```

```r
is.double(x1)
```

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

```r
is.atomic(x1)
```

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

```r
x1[1]
```

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

```r
x1[1:3]
```

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

```r
x1[c(1,2,3)]
```

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

```
x1[-2]
```

```
## [1]  3.0  5.0 12.0  3.4
```

```
x1[-c(1,2)]
```

```
## [1]  5.0 12.0  3.4
```

```
x1[3.12]
```

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

```
y1 = c(FALSE, 5L, 6.3, 'abc123')
typeof(y1)
```

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

```
str(y1)
```

```
##  chr [1:4] "FALSE" "5" "6.3" "abc123"
```

```
is.double(y1)
```

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

```
is.atomic(y1)
```

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

```
y1[3]
```

```
## [1] "6.3"
```

Funciones para manejo de vectores atómicos

```
(a  = c(9,4,2,3,4))
```

```
## [1] 9 4 2 3 4
```

```
(a1 = rep(4,3))
```

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

```
(a2 = seq(1,5))
```

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

```
(a3 = seq(1,5,0.25))
```

```
##  [1] 1.00 1.25 1.50 1.75 2.00 2.25 2.50 2.75 3.00 3.25 3.50 3.75 4.00 4.25 4.50
## [16] 4.75 5.00
```

```
(a4 = rev(a))
```

```
## [1] 4 3 2 4 9
```

```
(a5 = sort(a))
```

```
## [1] 2 3 4 4 9
```

```
(a6 = unique(a))
```

```
## [1] 9 4 2 3
```

```r
(a7 = letters)
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q" "r" "s"
## [20] "t" "u" "v" "w" "x" "y" "z"
```

```r
which(letters=="z")
```

```
## [1] 26
```

Lista

Se trata de una colección de n datos que pueden ser de distinto tipo. Puede incluir vectores atómicos dentro. Por ejemplo:

```r
x2 = list(1L, 'a')
typeof(x2)
```

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

```r
str(x2)
```

```
## List of 2
##  $ : int 1
##  $ : chr "a"
```

```r
is.atomic(x2)
```

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

```r
is.list(x2)
```

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

```r
x2[2]
```

```
## [[1]]
## [1] "a"
```

```r
x2[[2]]
```

```
## [1] "a"
```

```r
is.double(x2)
```

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

```r
is.integer(x2)
```

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

```r
is.character(x2)
```

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

```r
x3 = list(1L, 2L)
is.integer(x3)
```

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

```r
y2 = list(m = c(3,4,5), n = c(7,24,25))
typeof(y2)
```

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

```
str(y2)
```

```
## List of 2
##  $ m: num [1:3] 3 4 5
##  $ n: num [1:3] 7 24 25
```

```
is.atomic(y2)
```

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

```
is.list(y2)
```

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

```
y2[[2]]
```

```
## [1]  7 24 25
```

```
y2[[2]][3]
```

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

```
y2$n[3]
```

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

Funciones para manejo de listas

```
b1 = list(x = 4L, y = TRUE, 5-3i)
names(b1)
```

```
## [1] "x" "y" ""
```

```
unlist(b1)
```

```
##    x    y
## 4+0i 1+0i 5-3i
```

```
b2 = list(x = c("r","a","f"), y = TRUE, z = list(0,1))
names(b2)
```

```
## [1] "x" "y" "z"
```

```
unlist(b2)
```

```
##     x1     x2     x3      y     z1     z2
##    "r"    "a"    "f" "TRUE"    "0"    "1"
```

Matriz

Se trata de una colección de m x n datos del mismo tipo (Integer, Double, Logical, Character, Complex). De ser necesario, aplica coerción implícita. Por ejemplo:

```
x3 = matrix(c(7,3,5,8,6,6),ncol=2)
typeof(x3)
```

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

```
str(x3)
```

```
##  num [1:3, 1:2] 7 3 5 8 6 6
```

```r
is.double(x3)
```

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

```r
is.vector(x3)
```

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

```r
is.list(x3)
```

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

```r
is.matrix(x3)
```

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

```r
x3[1,2]
```

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

```r
x3[2,]
```

```
## [1] 3 6
```

```r
x3[,2]
```

```
## [1] 8 6 6
```

```r
x3[c(1,3),2]
```

```
## [1] 8 6
```

```r
x3[2:3,2]
```

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

```r
y3 = matrix(c(7,TRUE,5,8,1,2),nrow=3)
typeof(y3)
```

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

```r
str(y3)
```

```
##  num [1:3, 1:2] 7 1 5 8 1 2
```

```r
is.double(y3)
```

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

```r
is.vector(y3)
```

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

```r
is.list(y3)
```

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

```r
is.matrix(y3)
```

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

```r
y3[3,1]
```

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

```
y3[2,]
```

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

```
y3[,2]
```

```
## [1] 8 1 2
```

Funciones para manejo de matrices

```
c1 = matrix(c(5,6,3,4,3,3,1,1,8),ncol = 3, byrow = TRUE, dimnames = list(c("f1","f2","f3"),c("co1","co2
dim(c1)
```

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

```
rownames(c1)
```

```
## [1] "f1" "f2" "f3"
```

```
colnames(c1)
```

```
## [1] "co1" "co2" "co3"
```

```
t(c1)
```

```
##     f1 f2 f3
## co1  5  4  1
## co2  6  3  1
## co3  3  3  8
```

```
solve(c1)
```

```
##              f1          f2          f3
## co1 -0.31818182  0.68181818 -0.13636364
## co2  0.43939394 -0.56060606  0.04545455
## co3 -0.01515152 -0.01515152  0.13636364
```

```
c2 = matrix(c(3,4,3,4,6,7),nrow = 3)
cbind(c1,c2)
```

```
##    co1 co2 co3
## f1   5   6   3 3 4
## f2   4   3   3 4 6
## f3   1   1   8 3 7
```

```
c3 = matrix(c(3,4,3,4,6,7),ncol = 3)
rbind(c1,c3)
```

```
##    co1 co2 co3
## f1   5   6   3
## f2   4   3   3
## f3   1   1   8
##      3   3   6
##      4   4   7
```

Data frame

Se trata de una colección de m x n datos donde cada columna puede ser de distinto tipo. Es la estructura comúnmente utilizada para trabajar con conjuntos de datos reales. Por ejemplo:

```r
var1 = c(4,5,3,5,3)
var2 = c('1','3','66','12','15')
x4 = data.frame(var1, var2)
typeof(x4)
```

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

```r
str(x4)
```

```
## 'data.frame':    5 obs. of  2 variables:
##  $ var1: num  4 5 3 5 3
##  $ var2: chr  "1" "3" "66" "12" ...
```

```r
is.double(x4)
```

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

```r
is.vector(x4)
```

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

```r
is.list(x4)
```

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

```r
is.matrix(x4)
```

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

```r
is.data.frame(x4)
```

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

```r
x4[1,2]
```

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

```r
x4[2,]
```

```
##   var1 var2
## 2    5    3
```

```r
x4[,2]
```

```
## [1] "1"  "3"  "66" "12" "15"
```

```r
x4$var1
```

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

```r
x4$var2[3]
```

```
## [1] "66"
```