3. Estructuras de datos

Índice

1. Vectores

Un vector es una colección de elementos que son del mismo tipo:

```
numbers <- c(1, 2, 3, 4, 5)
numbers2 <- 1:5
numbers3 <- 1.5:5.5
numbers4 <- 1.5:5.3
```

Se puede acceder a un elemento por su índice (los vectores empiezan en 1):

```
numbers[1]
numbers[3] <- 7
```

Obtener un subconjunto usando otro vector como índice, o un índice negativo:

```
numbers[c(1, 3)]
numbers[c(-1)]
```

Existen algunas funciones útiles como conocer la longitud, ordenar el vector o hacer copias:

```
length(numbers)
sort(numbers)

numbers5 <- rep(c(1,2,3), each = 3)
numbers6 <- rep(c(1,2,3), times = 3)
numbers7 <- rep(c(1,2,3), times = c(5,2,1))
```

Para generar vectores con secuencias de números no correlativos:

```
numbers8 <- seq(from = 0, to = 100, by = 20)
```

2. Listas

Una lista es una colección de elementos que pueden ser de distintos tipos:

```
months <- list("enero", 2, "marzo", TRUE)
```

Tienen algunas cosas en común con los vectores como acceso, longitud, etc.

Para comprobar que un elemento está contenido en una lista:

```
"febrero" %in% months
```

Para añadir elementos a la lista, bien por el final o en una posición concreta:

```
append(months, "octubre")
append(months, "abril", after = 3)
```

Para eliminar elementos, se debe generar una nueva lista subconjunto de la primera y reasignar:

```
months <- months[-1]
months <- months[2:3]
```

Se pueden unir listas:

```
moreMonths <- list(3.14, "diciembre")
totalMonths <- c(totalMonths, moreMonths)
```

O iterar sobre ellas con un bucle:

```
for (m in totalMonths) {
  print(m)
}
```

3. Matrices

Las matrices son conjuntos bidimensionales de datos del mismo tipo, con filas y columnas:

```
numbers <- matrix(c(1,2,3,4,5,6), nrow = 3, ncol = 2)
```

Tienen algunas cosas en común con las listas como el operador `%in%` o la longitud, siendo esta la cantidad total de elementos. Para saber las dimensiones:

```
dim(numbers)
```

Se puede acceder a un elemento en una posición concreta, obtener toda una fila o columna completa, o incluso más de una fila o columna:

```
numbers[2,2]
numbers[2,]
numbers[,2]
numbers[c(1,2),]
numbers[,c(1,2)]
numbers[c(1,2),c(1,2)]
```

Para añadir filas o columnas a la matriz, o combinar matrices:

```
numbers <- rbind(numbers, c(7,8))
numbers <- cbind(numbers, c(9,10,11))

moreNumbers <- matrix(c(21,22,23,24,25,26), nrow = 2, ncol = 3)
numbers <- rbind(numbers, moreNumbers)
```

Para eliminar filas o columnas, se genera una matriz subconjnto para reasignar:

```
numbers <- numbers[-c(1), -c(1)]
```

Se pueden iterar con dos bucles anidados (uno para filas y otro para columnas):

```
for (i in 1:nrow(numbers)) {
  for (j in 1:ncol(numbers)) {
    print(numbers[i, j])
  }
}
```

4. Data Frames

Los data frames son conjuntos bidimensionales de datos que pueden ser de distintos tipos por columnas, útiles para representar tablas:

```
students <- data.frame (
  Name = c("Ada Lovelace", "Miyamoto Mushashi", "Siddartha Gautama", "Clara
Campoamor"),
  Age = c(32, 41, 28, 17),
  Grade = c(8, 3, 7, 5)
)
```

Para obtener un resumen del detalle del data frame:

```
summary(students)
```

Se puede acceder a columnas completas de diversas formas:

```
students[1]
students[["Name"]]
students$Name
```

Tienen algunas cosas en común con las matrices como añadir o borrar filas o columnas, saber su longitud total, dimensiones o longitud por fila o columna, o combinar data frames entre sí.

Resumen comparativo

	Homogeneo	**Heterogeneo**
1d	Vector	Lista
2d	Matriz	Data frame
nd	Array	

Referencias

Vectores
Listas
Matrices
Data Frames
Arrays
Factors