Vectores

HR Analytics: Teoría y Práctica

http://pablohaya.com/contact

09/2018

Vectores

Permiten almacenar secuencias de valores del mismo tipo en una misma variable. La función `c()` permite combinar varios datos para crear un vector.

```
v <- c(1, 1, 2, 3, 5, 8)
v
```

```
## [1] 1 1 2 3 5 8
```

Es importante mencionar que no es posible almacenar distinos tipos de datos en un mismo vector. En caso de crear un vector de datos de distinto tipo, automáticamente se convierten para que todos compartan el mismo tipo.

```
v <- c(1, "uno", 2, "tres", 5, "ocho")
v
```

```
## [1] "1"    "uno"  "2"    "tres" "5"    "ocho"
```

```
v <- c(FALSE, TRUE, 2, 3, 5, 8)
v
```

```
## [1] 0 1 2 3 5 8
```

`c()` permite también combinar dos o más vectores en un sólo vector

```
v1 <- c(1, 2)
v2 <- c(3, 4)
v3 <- c(v1, v2)
v3
```

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

Secuencias

Otra función que permite crear un vector es `seq()` que crea una secuencia de números. Esta función, en su versión más simple, se puede escribir también como `desde:hasta`

```
seq(0, 10)
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10
```

```
0:10
```

```
##  [1]  0  1  2  3  4  5  6  7  8  9 10
```

```r
seq(0, 10, by=2)
```

```
## [1]  0  2  4  6  8 10
```

```r
seq(0, 1, by=0.1)
```

```
##  [1] 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0
```

Repeticiones

De manera similar la función `rep()` replica un elemento tantas veces como le digamos.

```
rep(10, 5)
```

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

```
rep(1:4, 2)
```

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

```
rep(1:4, each = 2)
```

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

Indexación

Existen múltiples maneras de poder acceder a los elementos de vector. La más sencilla es mediente su posición indicada entre corchetes `ej. vector[2]`.

```
v <- c(1, 2, 4, 8, 16, 32)
v[1]
```

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

```
v[1] + v[2]
```

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

```
v[6]
```

```
## [1] 32
```

La función `length()` nos permite obtener la longitud de un vector, la cual concide con la última posición del mismo.

```
v <- c(1, 2, 4, 8, 16, 32)
length(v)
```

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

```
v[length(v)]
```

```
## [1] 32
```

Nombres

Es posible asignar nombres a cada posición del vector (`names()`), de manera que luego se puede emplear el nombre, en vez de la posición, para referenciar a un elemento.

```
v <- c(34, 23, 12, 14, 23)
names(v) <- c("L", "M", "X", "J", "V")
v
```

```
##  L  M  X  J  V
## 34 23 12 14 23
```

```
v["X"]
```

```
##  X
## 12
```

Subconjuntos

A partir de un vector es posible acceder a un subconjunto del mismo indicando el vector de posiciones que queremos obtener.

```
v <- c(10, 30, 50, 70, 90)
v[c(1, 3, 5)]
```

```
## [1] 10 50 90
```

```
v[2:3]
```

```
## [1] 30 50
```

Se puede indicar que posiciones **no** queremos mediante índices negativos.

```
v <- c(10, 30, 50, 70, 90)
v[c(-1, -3, -5)]
```

```
## [1] 30 70
```

Al igual que veíamos antes, si las posiciones tienen nombres se pueden acceder mediante un vector de nombres a varias posiciones.

```
v <- c(34, 23, 12, 14, 23)
names(v) <- c("L", "M", "X", "J", "V")
v[c("M", "J")]
```

```
##  M  J
## 23 14
```

Finalmente, se puede especificar un vector de valores boolenos para indicar qué elementos queremos y cuales no.

```
v <- c(10, 30, 50, 70, 90)
v[c(FALSE, TRUE, FALSE, TRUE, FALSE)]
```

```
## [1] 30 70
```

Vectorización de operaciones

Las operaciones que hagamos entre dos vectores se realizan elemento a elemento.

```
v1 <- c(1, 2, 3, 4, 5)
v2 <- c(10, 20, 30, 40, 50)
v1 + v2
```

```
## [1] 11 22 33 44 55
```

```
v1 * v2
```

```
## [1]  10  40  90 160 250
```

Si los vectores tienen distinta longitud se rellena el vector de menor longitud repetiendo sus elementos hasta llegar al tamaño del otro vector.

```
v1 <- c(1, 2)
v2 <- c(10, 20, 30, 40, 50, 60)
# v1 se amplica a c(1, 2, 1, 2, 1, 2)
v1 * v2
```

```
## [1]  10  40  30  80  50 120
```

En particular, si operamos entre un número y vector, el resultado es la operación de cada elemento del vector con el número.

```
x <- 2
v <- c(1, 2, 3, 4, 5, 6)
v ** x
```

```
## [1]  1  4  9 16 25 36
```

Esto también funciona como operadores de comparación siendo el resultado un vector de valores lógicos.

```
v <- c(1, 2, 3, 4, 5, 6)
v < 4
```

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

Lo cual es bastante interesante ya que se puede emplear el vector resultante de la comparación para extraer el subconjunto de valores del vector original que cumplen cierta condición.

```
v <- c(1, 2, 3, 4, 5, 6)
# recordar que v < 4
# devuelve c(TRUE, TRUE, TRUE, FALSE, FALSE, FALSE)
v[v < 4]
```

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

```
v[c(TRUE, TRUE, TRUE, FALSE, FALSE, FALSE)]
```

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

Funciones que resumen un vector

Existen funciones que emplean todos los elementos de un vector para obtener un único valor (ej. `sum`, `mean`, `min`, `max`...)

```
v <- c(1, 2, 3, 4, 5, 6)
sum(v)
```

```
## [1] 21
```

```
min(1:6)
```

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

```
mean(v)
```

```
## [1] 3.5
```