**LISTAS**

**¿Qué es una Lista?**

Una lista es una estructura de datos, muy parecida a un vector, en el sentido de que se utiliza para almacenar un conjunto ordenado de elementos.

Sin embargo, cuando un vector requiere que todos sus elementos sean del mismo tipo, una lista permite recopilar diferentes tipos de elementos.

Debido a esta flexibilidad, las listas se utilizan a menudo para almacenar diversos tipos de datos de entrada y salida y conjuntos de parámetros de configuración para modelos de aprendizaje automático.

**Ilustrando**

Para ilustrar listas, considere el conjunto de datos de pacientes médicos que hemos estado construyendo con los datos de tres pacientes almacenados en seis vectores.

Si queremos mostrar todos los datos en John Doe (sujeto 1), deberíamos ingresar cinco comandos en R:

Ingresemos los 6 vectores.

subject\_name **<-** c**(**"John Doe", "Jane Doe", "Steve Graves"**)**

temperature **<-** c**(**98.1, 98.6, 101.4**)**

flu\_status **<-** c**(FALSE**, **FALSE**, **TRUE)**

gender **<-** factor**(**c**(**"MALE", "FEMALE", "MALE"**))**

blood **<-** factor**(**c**(**"O", "AB", "A"**)**, levels **=** c**(**"A", "B", "AB", "O"**))**

symptoms **<-** factor**(**c**(**"SEVERE", "MILD", "MODERATE"**)**,levels **=** c**(**"MILD", "MODERATE", "SEVERE"**)**, ordered **=** **TRUE)**

Busquemos los datos que se encuentran con el vector 1.

subject\_name**[**1**]**;temperature**[**1**]**;flu\_status**[**1**]**;gender**[**1**]**;blood**[**1**]**;symptoms**[**1**]**

Resultado:

[1] "John Doe"

[1] 98.1

[1] FALSE

[1] MALE

Levels: FEMALE MALE

[1] O

Levels: A B AB O

[1] SEVERE

Levels: MILD < MODERATE < SEVERE

Esto parece mucho trabajo para mostrar los datos médicos de un paciente. La estructura de la lista nos permite agrupar todos los datos del paciente en un objeto que podemos usar repetidamente.

**Creando una lista**

Similar a crear un vector con c (), una lista se crea utilizando la función list (), como se muestra en el siguiente ejemplo.

Una diferencia notable es que cuando se construye una lista, a cada componente de la secuencia casi siempre se le asigna un nombre.

Los nombres no son técnicamente necesarios, pero permiten acceder a los valores de la lista más adelante por nombre en lugar de por posición numerada.

Para crear una lista con componentes con nombre para todos los datos del primer paciente, escriba lo siguiente:

subject1 **<-** list**(**fullname **=** subject\_name**[**1**]**, temperature **=** temperature**[**1**]**, flu\_status **=** flu\_status**[**1**]**,gender **=** gender**[**1**]**,blood **=** blood**[**1**]**, symptoms **=** symptoms**[**1**])**

Posteriormente solo llame al objeto para obtener los datos.

**>** subject1

**$**fullname

**[**1**]** "John Doe"

**$**temperature

**[**1**]** 98.1

**$**flu\_status

**[**1**]** **FALSE**

**$**gender

**[**1**]** MALE

Levels**:** FEMALE MALE

**$**blood

**[**1**]** O

Levels**:** A B AB O

**$**symptoms

**[**1**]** SEVERE

Levels**:** MILD **<** MODERATE **<** SEVERE

**Tenga en cuenta**

Que los valores están etiquetados con los nombres que especificamos en el comando anterior. Sin embargo, todavía se puede acceder a una lista utilizando métodos similares a un vector.

Para acceder al valor de temperatura, use el siguiente comando:

subject1**[**2**]**

**Operador [[]]**

El resultado de usar operadores de estilo vectorial en un objeto de lista es otro objeto de lista, que es un subconjunto de la lista original.

Por ejemplo, el código anterior devolvió una lista con un solo componente de temperatura.

Para devolver un solo elemento de la lista en su tipo de datos nativo, use corchetes dobles ([[y]]) cuando intente seleccionar el componente de la lista.

Por ejemplo, lo siguiente devuelve un vector numérico de longitud uno:

subject1**[[**2**]]**

**>** subject1**[[**2**]]**

**[**1**]** 98.1

**Operador $**

Para mayor claridad, a menudo es más fácil acceder a los componentes de la lista directamente, agregando $ y el nombre del valor al nombre del componente de la lista, de la siguiente manera:

subject1**$**temperature

**[**1**]** 98.1

Al igual que la notación de doble corchete, esto devuelve el componente de lista en su tipo de datos nativos (en este caso, un vector numérico de longitud uno).

**Acceso al Valor por Nombre**

El acceso al valor por nombre también garantiza que se recupere el elemento correcto, incluso si el orden de los elementos de la lista se modifica más adelante.

Es posible obtener varios elementos en una lista especificando un vector de nombres. Lo siguiente devuelve un subconjunto de la lista del sujeto 1, que contiene solo los componentes de temperatura y estado de la gripe:

subject1**[**c**(**"temperature", "flu\_status"**)]**

**Conclusión**

Conjuntos de datos completos se podrían construir utilizando listas y listas de listas.

Por ejemplo, podría considerar crear una lista subject2 y subject3, y combinarlas en un solo objeto de lista llamado pt\_data.

Sin embargo, la construcción de un conjunto de datos de esta manera es lo suficientemente común como para que R proporcione una estructura de datos especializada específicamente para esta tarea, los cuales veremos más adelante.

**EJERCICIOS**

**Introducción:** En R la lista es el objeto que contiene elementos de diferentes tipos, como cadenas, números, vectores y otra lista dentro de él. La lista de R también puede contener una matriz o una función como sus elementos. La lista se crea usando la función list () en R. En otras palabras, una lista es un vector genérico que contiene otros objetos.

**Ejercicio 1.**

Si

p <- c(2,7,8)

q <- c("A", "B", "C")

x <- list(p, q)

¿Cuál es el valor de x[2]?

a. NULL

b. "A" "B" "C"

c. "7"

**Ejercicio 2.**

Si

w <- c(2, 7, 8)

v <- c("A", "B", "C")

x <- list(w, v)

Como reemplazaría “A” en x con “K”

a. x[[2]] <- "K"

b. x[[2]][1] <- "K"

c. x[[1]][2] <- "K"

**Ejercicio 3.**

Si

a <- list ("x"=5, "y"=10, "z"=15)

¿Qué escribiría para la suma de todos los elementos en a?

a. sum(a)

b. sum(list(a))

c. sum(unlist(a))

**Ejercicio 4.**

Si

Newlist <- list(a=1:10, b="Good morning", c="Hi")

Escriba una instrucción en R para agregar 1 elemento al primer vector de Newlist.

**Ejercicio 5.**

Si

b <- list(a=1:10, c="Hello", d="AA")

Escriba una expresión en R que contenga todos los elementos, exceptuando el segundo del primer vector de b.

**Ejercicio 6.**

Si

x <- list(a=5:10, c="Hello", d="AA")

Escriba una declaración en R para agregar un nuevo item, z = “NewItem” en la lista x.

**Ejercicio 7.**

Considere lo siguiente:

y <- list("a", "b", "c")

Escriba una declaración en R que asigne nuevos nombres, “one”, “two”, y “three”, a los elementos de y.

**Ejercicio 8.**

Si

x <- list(y=1:10, t="Hello", f="TT", r=5:20)

Escriba una declaración en R que muestre la longitud del vector r en x.

**Tip:** Busque y utilice la función length(), adecuadamente.

**Ejercicio 9.**

Si

string <- "Grand Opening"

Escriba una declaración en R que retorne la siguiente salida:

[[1]]

[1] "Grand"

[[2]]

[1] "Opening"

**Tip:** Busque la función strsplit() para efectuar el ejercicio.

**Ejercicio 10.**

Si

y <- list("a", "b", "c")

q <- list("A", "B", "C", "a", "b", "c")

Escriba una declaración que retorne todos los elementos de q, que no estén en y, con el resultado siguiente:

[[1]]

[1] "A"

[[2]]

[1] "B"

[[3]]

[1] "C"

**Tip:** Utilice la función setdiff() para la solución.