uc3m | Universidad Carlos III de Madrid

Programación II

Grado en Estadística y Empresa

Estructuras de Datos

Autores

Dr. Fuensanta Medina Domínguez
Dr. María Isabel Sánchez Segura
Dr. Antonio de Amescua Seco

CC BY NC SA

Estructuras de Datos: Vectores

- Se pueden definir vectores numéricos, lógicos y de texto: *c(elementos)*

```
>c(1,5,3,4)
 [1]1 5 3 4

>c(T,F,T,F,T)    #vector lógico de 5 elementos
 [1] TRUE  FALSE  TRUE  FALSE  TRUE

>c("madrid", "valencia", "getafe") #vector con 3 cadenas de texto
 [1]  madrid  valencia  getafe
```

Vectores

- La letra *c* concatena dos vectores o elementos a un vector:

```
>x=c(1,3,5)
>y=c(2,4,6)
>c(x,y)
[1] 1  3  5  2  4  6
```

- Para eliminar elemento del vector: *nombreVector[-posicion]*

```
>x=c(1,3,5,2,4,6)
>x[-4]  #elimina el elemento de la posición 4
[1] 1  3  5  4  6
>x[-c(1,3)] #elimina el elemento de la posición 1 y 3
[1]3  2  4  6
```

Vectores

- Si queremos seleccionar elementos de un vector: *nombreVector[posición]*

```
>x=c(1,3,5,2,4,6)
>x[c(1,3,6)]
[1] 1  5  6
```

- Podemos especificar una condición lógica:

```
>x=c(1,3,5,2,4,6)
>x>3
[1] FALSE  FALSE  TRUE  FALSE  TRUE  TRUE
>x[x>3]
[1] 5  4  6
```

Vectores: Funciones

> x=c(1,3,5,2,4,6)

sum(x) **[1] 21**	**#suma los elementos de un vector**
min(x) [1]1	#devuelve el elemento más pequeño del vector
max(x) [1] 6	#devuelve el elemento mayor del vector
length(x) [1] 6	#devuelve el número de elementos de un vector
range(x) [1] 1 6	#devuelve el rango del vector
diff(x) [1] 2 2 -3 2 2	#devuelve la diferente entre elementos consecutivos
mean(x) [1]3.5	#devuelve la media de los elementos del vector
sort(x) [1]1 2 3 4 5 6	#ordena los elementos del vector de menor a mayor
cumsum(x) [1]1 4 9 11 15 21	# suma acumulada de elementos consecutivos

Vectores: Operadores Lógicos

- Vectores lógicos se crean usando c() o es la respuesta a una condición

- Operadores lógicos:

```
>x=1:5
>x < 5                          #x menor que 5
[1]TRUE TRUE TRUE TRUE FALSE
>x>1                            #x mayor que 1
[1] FALSE TRUE TRUE TRUE TRUE
>x>1 & x<5                      #x mayor que 1 Y menor que 5
[1] FALSE TRUE TRUE TRUE FALSE
x>1 && x<5                      #comprueba el primer elemento – Y lógico
[1] FALSE
>x>1 | x<5                      #x mayor que 1 O menor que 5
[1] TRUE TRUE TRUE TRUE TRUE
```

Vectores: Operadores Lógicos

```
>x=1:5
x>1 || x<5                                  #comprueba el primero
[1] TRUE
>x ==3                                  #x igual a 3
[1] FALSE FALSE TRUE FALSE FALSE
>x!=3                                       #x distinto a 3
[1] TRUE TRUE FALSE TRUE TRUE
>! x==3                                 #no (x igual a 3)
[1] TRUE TRUE FALSE TRUE TRUE
>y=2:5
>identical(x,y)
[1] FALSE
```

Estructura de Datos: Matriz

- Matriz: conjunto de objetos indizados por filas y columnas

- Crear una matriz: *matrix(data,nrow,ncol,byrow)*
  - data: los elementos que forman la matriz. Tienen que ser **todos del mismo tipo** (enteros, carácter, lógico), no pueden estar mezclados.
  - nrow: número de filas de la matriz
  - ncol: número de columnas de la matriz
  - byrow: los datos se colocan por filas o columnas según se van leyendo. Por defecto los datos se colocan por columnas.

Matriz

EJEMPLOS	
> matrix(1:4) [,1] [1,] 1 [2,] 2 [3,] 3 [4,] 4	> matrix(1:6, nrow=2) [,1] [,2] [,3] [1,] 1 3 5 [2,] 2 4 6
> matrix(1:6,nrow=3) [,1] [,2] [1,] 1 4 [2,] 2 5 [3,] 3 6	> matrix(1:6, nrow=2,byrow=T) [,1] [,2] [,3] [1,] 1 2 3 [2,] 4 5 6

Matriz: Funciones

- Funciones:

- length(x): devuelve número total de elementos de la matriz x
- mode(x): tipo de datos de los elementos de la matriz x
- dim(x): devuelve las dimensiones de la matriz x
- dimnames(x): devuelve los nombres de las dimensiones de la matriz x
- rownames(x): nombre de las filas de la matriz x
- colnames(x): nombre de las columnas de la matriz x
- is.matrix(x): devuelve si el objeto x es una matriz

Matriz: Ejemplos

```
> x=matrix(1:6, nrow=3)
> x
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

> cbind(x,c(10,10,10))	[,1] [,2] [,3] [1,] 1 4 10 [2,] 2 5 10 [3,] 3 6 10
> rbind(x,c(20,20))	[,1] [,2] [1,] 1 4 [2,] 2 5 [3,] 3 6 [4,] 20 20
> x[1,] #Primera fila	[1] 1 4
> x[,1] #Primera columna	[1] 1 2 3
> x[2,2]	[1] 5

Matriz: Ejemplos

```
mat=matrix(c(20,65,1.74,22,70,1.80,19,68,1.70),nrow=3,byrow=T)
mat
     [,1] [,2]  [,3]
[1,]   20   65 1.74
[2,]   22   70 1.80
[3,]   19   68 1.70
```

> colnames(mat)=c("edad","peso","altura") #añade nombre a las columnas	>mat edad peso altura [1,] 20 65 1.74 [2,] 22 70 1.80 [3,] 19 68 1.70
> rownames(mat)=c("juan","maria","ana") #añade nombre a las filas	> mat edad peso altura juan 20 65 1.74 maria 22 70 1.80 ana 19 68 1.70
> mat["juan",]	edad peso altura 20.00 65.00 1.74
> mat[,"edad"]	juan maria ana 20 22 19
> mat[,c("edad","altura")]	edad altura juan 20 1.74 maria 22 1.80 ana 19 1.70

Estructura de datos: Factor

- Un factor es un vector que se usa para especificar una clasificación discreta de los componentes de otros vectores de la misma longitud.

```
> students.origin=c("londres", "paris", "madrid", "madrid" , "paris", "roma")
> fstudents=as.factor(students.origin)
> fstudents
[1] londres paris   madrid  madrid  paris   roma
Levels: londres madrid paris roma

> summary(fstudents)
londres  madrid   paris    roma
      1       2       2       1
```

Factor

- Para saber el nombre de los niveles:

```
> Data<-factor(c(“mujer”, “hombre”, “mujer”))
> levels(Data)
 [1] "hombre" "mujer"
> nlevels(Data)
 [1] 2
> length(levels(Data))
[1] 2
```

Por defecto, los niveles de los factores son tratados en orden alfabético

Estructuras de Datos: Listas

- Listas sirven para concatenar objetos donde cada uno puede tener una estructura distinta.
- Una lista tiene una serie de componentes, a los que se debe asignar un nombre.

```
> alumno=list(nia="1000001",nombre="Francisco Medina", dni="44144232D")
> alumno
        $nia
        [1] "1000001“
        $nombre
        [1] "Francisco Medina“
        $dni
        [1] "44144232D"
```

Listas

- Para ver los nombres de los objetos dentro de la lista se usa $ seguido del nombre del componente o *[[nº del componente]]*

```
> names(alumno)
[1] "nia"    "nombre"  "dni"
```

Para acceder a componentes concretos:

```
> alumno$dni
[1] " 44144232D“

> alumno[[3]]
[1] " 44144232D "
```

Estructuras de Datos: Data Frames

- Data Frames: estructura de datos que generaliza a las matrices:
  - las columnas pueden ser de **diferente tipo de datos entre sí**.
  - todos los elementos de una misma columna deben ser del mismo tipo.
  - todos los elementos deben ser de la misma longitud.

Data Frames: Ejemplo

- Si se crea una matriz:

```
>datos=matrix(c(7.5,7.2,6.5,7.0,8.5,8.5,5.5,6.0,7.5,7.0,8.0,8.0),nrow=6,byrow=T)
> datos
     [,1] [,2]
[1,]  7.5  7.2
[2,]  6.5  7.0
[3,]  8.5  8.5
[4,]  5.5  6.0
[5,]  7.5  7.0
[6,]  8.0  8.0
```

Data Frames

- Para renombrar las filas y columnas: *dimnames()*

```
>dimnames(datos)=list(c("ana","pepe","nacho","bea","gema","alba"),
    c("Matematicas","Fisica"))

> datos
      Matematicas Fisica
ana           7.5    7.2
pepe          6.5    7.0
nacho         8.5    8.5
bea           5.5    6.0
gema          7.5    7.0
alba          8.0    8.0
```

Data Frames: Ejemplo

- Si se añade una columna a la matriz de datos de diferente tipo hay que convertir la matriz en dataframe:

```
> datos2=data.frame(datos, provincia)
> datos2
       Matematicas  Fisica  provincia
ana            7.5     7.2     madrid
pepe           6.5     7.0       leon
Nacho          8.5     8.5     oviedo
bea            5.5     6.0     malaga
gema           7.5     7.0    sevilla
alba           8.0     8.0     madrid

> mean(datos2[,"Matematicas"])
[1] 7.25
```

Data Frames

- Para acceder a los datos del dataframe se realiza como matriz o lista:

```
> datos2[,"Fisica"]
[1] 7.2 7.0 8.5 6.0 7.0 8.0

> datos2$Fisica
[1] 7.2 7.0 8.5 6.0 7.0 8.0
```

Estructuras de Datos: Array

- Array: generalización de una matriz al caso multidimensional.
- Crear un array: *array(datos, dimensiones)*

```
> array(1:12,c(2,3,2))
, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
```

Array:Ejemplos

```
> array(1:12,c(2,3,3))
, , 1
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
, , 2
     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
, , 3
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Array: Ejemplos

```
>x=array(c(75,72,65,70,85,85,55,60,75,70,80,80),c(2,3,2) )
>dimnames(x) = list(c("hombres","mujeres"),c(“Estadistica",”Fisica“, “Programacion"),
    c("getafe", "leganes"))
```

```
> x
, , getafe
                Estadistica   Fisica   Programacion
hombres         75            65       85
mujeres         72            70       85

, , leganes
                Estadistica   Fisica   Programacion
hombres         55            75       80
Mujeres         60            70       80
```

Array: Ejemplos

```
> dimnames(x)      #nombre de las dimensiones del array
[[1]]
[1] "hombres" "mujeres"

[[2]]
[1] "Estadistica"  "Fisica"       "Programacion"

[[3]]
[1] "getafe"  "leganes"
```

Array

- Acceder a los datos del campus de Leganés:

```
> x[,,"leganes"]
           Estadistica Fisica Programacion
hombres             55     75          80
mujeres             60     70          80
```

- Acceder a los datos de todos los hombres:

```
> x["hombres",,]
            getafe leganes
Estadistica     75      55
Fisica          65      75
Programacion    85      80
```