Estadística Computacional:

Tercera Clase

Andrés Aravena

Vectores

- Todos los elementos del vector deben tener el mismo tipo
- En caso de mezcla se convierten al tipo más genérico

```
> c(1, “sello”)
  [1] "1"     "sello"

> c(TRUE, “sello”)
  [1] "TRUE"  "sello"
```

Combinaciones

```
> c(2,TRUE, FALSE)
  [1] 2 1 0

> c(factor(c("a","b")),"c")
  [1] "1" "2" "c"
```

Datos Faltantes

- En la práctica hay casos en que un dato no está
- No conviene inventar un valor ficticio
- El símbolo NA representa ese caso
- Se puede usar en cualquier vector, independiente del tipo

```
> c(NA,TRUE, FALSE)
[1]    NA  TRUE FALSE
> c(NA,1,2)
[1] NA  1  2
```

Creando vectores

- Concatenación simple

```
> x <- c(1,2,3)

> y <- c(10,20)

> c(x, y, 5)
 [1]  1  2  3 10 20  5
```

Creando vectores

- Vectores lógicos

```
> c(TRUE, TRUE, FALSE, TRUE)
  [1]  TRUE  TRUE FALSE  TRUE
```

- También se puede escribir `c(T,T,F,T)`

```
> IMC>25
 [1] FALSE FALSE FALSE FALSE  TRUE FALSE
```

Secuencias

```
> 4:9
  [1] 4 5 6 7 8 9

> seq(4,9)
  [1] 4 5 6 7 8 9

> seq(4,10,2)
  [1]  4  6  8 10

> seq(from=4, by=2, length=4)
  [1]  4  6  8 10
```

Repeticiones

```
> rep(1,3)
  [1] 1 1 1

> rep(c(7,9,13), 3)
  [1]  7  9 13  7  9 13  7  9 13

> rep(c(7,9,13), 1:3)
  [1]  7  9  9 13 13 13
```

Repeticiones

```
> rep(1:2,c(10,5))
 [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2

> rep(c(TRUE,FALSE),3)
 [1]  TRUE FALSE  TRUE FALSE  TRUE FALSE

> rep(c(TRUE,FALSE),c(3,3))
 [1]  TRUE  TRUE  TRUE FALSE FALSE FALSE
```

Nombres

- Cada elemento puede tener un nombre

```
> peso <- c(Pedro=60, Juan=72, Diego=57,
  Hugo=90, Paco=95, Luis=72)

> names(peso)
  [1] "Pedro" "Juan" "Diego" "Hugo" "Paco"
  "Luis"

> names(altura) <- names(peso)
```

Listas

- Como vectores, pero con elementos de distinto tipo

```
personas <- list(peso=c(60,72,57,90,95, 72),
altura=c(1.75,1.80,1.65,1.90,1.74, 1.91),
nombres=c("Pedro","Juan","Diego","Hugo","Paco",
"Luis"),
valido=TRUE,
genero=factor(rep("H",6),levels=c("H","M")))
```

```
> personas
$peso
[1] 60 72 57 90 95 72

$altura
[1] 1.75 1.80 1.65 1.90 1.74 1.91

$nombres
[1] "Pedro" "Juan"  "Diego" "Hugo"  "Paco"  "Luis"

$valido
[1] TRUE

$genero
[1] H H H H H H
Levels: H M
```

Matrices

- Como vectores pero en 2 dimensiones

```
> matrix(peso, nrow=2, ncol=3)
     [,1] [,2] [,3]
[1,]   60   57   95
[2,]   72   90   72

> matrix(peso, nrow=2, ncol=3, byrow=T)
     [,1] [,2] [,3]
[1,]   60   72   57
[2,]   90   95   72
```

Matrices

```
> M=matrix(peso, nrow=2, ncol=3)
> dim(M)
  [1] 2 3
```

- Ver también `nrow(M) y ncol(M)`

```
> colnames(M) <- c("A","B","C")
> rownames(M) <- c("x","y")
> M
    A  B  C
  x 60 57 95
  y 72 90 72
```

Arreglos

- Como matrices pero en más dimensiones

```
> A=array(0, dim=c(2,3,2))

> A
, , 1

     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0

, , 2

     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
```

Data Frames

- Bidimensionales, parecen matrices
- Cada columna es de un tipo distinto

```
> pers <- data.frame(peso=c(60,72,57,90,95, 72),
  altura=c(1.75,1.80,1.65,1.90,1.74, 1.91),
  nombres=c("Pedro","Juan","Diego","Hugo","Paco",
  "Luis"),
  IMC=IMC,
  genero=factor(rep("H",6),levels=c("H","M")))
```

Data Frame

```
> pers
  peso altura nombres      IMC genero
1   60   1.75   Pedro 19.59184      H
2   72   1.80    Juan 22.22222      H
3   57   1.65   Diego 20.93664      H
4   90   1.90    Hugo 24.93075      H
5   95   1.74    Paco 31.37799      H
6   72   1.91    Luis 19.73630      H
```

Accediendo a elementos

- Para obtener el i-ésimo elemento de un vector usamos [i]

```
> peso[3]
  Diego
     57
> peso[c(1,3,5)]
  Pedro Diego  Paco
     60    57    95
> peso[2:4]
```

- El índice puede ser un vector de enteros

Índices Negativos

- Indican elementos a omitir

```
> peso
Pedro  Juan Diego  Hugo  Paco  Luis
   60    72    57    90    95    72
> peso[c(-1,-3,-5)]
Juan Hugo Luis
  72   90   72
```

- Útil cuando se usan casi todos los elementos

Índices Lógicos

- Se puede indexar por un vector lógico
- Debe ser del mismo largo del vector

```
> peso>72
   Pedro  Juan Diego  Hugo  Paco  Luis
   FALSE FALSE FALSE  TRUE  TRUE FALSE
> peso[peso>72]
   Hugo Paco
     90   95
```

Nombres como Índices

- Si el vector tiene nombres se puede usar:

```
> peso[c("Pedro","Juan","Diego")]
  Pedro  Juan Diego
     60    72    57
```

- ¿cómo saber si el vector tiene nombres?:
  - `!is.null(names(peso))`
  - No sirve: `names(peso) != NULL`

Indexando Matrices

- Objetos tipo `matrix`, `array` y `data.frame` usan un índice por cada dimensión
- Ej: `M[1,1], M["x","A"]`
- Si se omite un índice, se toma todo el rango

```
>  M[2,]
   A  B  C
  72 90 72
>  M[,3]
   x  y
  95 72
>  M[,2:3]
     B  C
  x 57 95
  y 90 72
```

Indexando Listas

- Se pueden indexar como vectores
- Retorna una sub lista

```
> personas[1:2]
$peso
[1] 60 72 57 90 95 72

$altura
[1] 1.75 1.80 1.65 1.90 1.74 1.91
```

```
> personas
$peso
[1] 60 72 57 90 95 72

$altura
[1] 1.75 1.80 1.65 1.90 1.74 1.91

$nombres
[1] "Pedro" "Juan"  "Diego" "Hugo"  "Paco"  "Luis"

$valido
[1] TRUE

$genero
[1] H H H H H H
Levels: H M
```

Elementos de Listas

```
> personas[1]
 $peso
 [1] 60 72 57 90 95 72
```

- Es una sublista

```
> personas[[1]]
 [1] 60 72 57 90 95 72
```

- Es un elemento
- Equivale a `personas[["peso"]]`
- O también a `personas$peso`