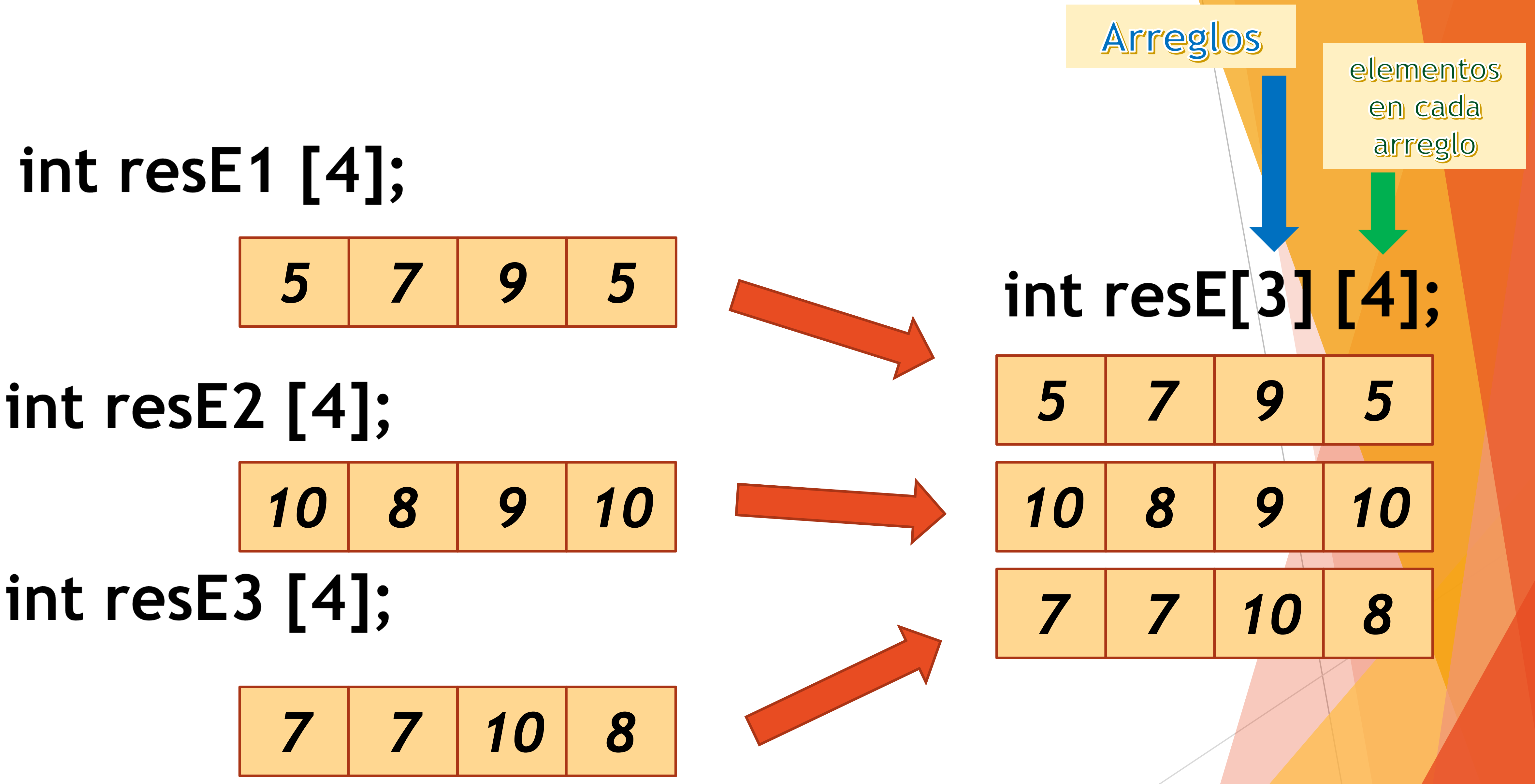
matrices



```
int resE1 [4];
```

5	7	9	5

```
int resE2 [4];
```

10	8	9	10

```
int resE3 [4];
```

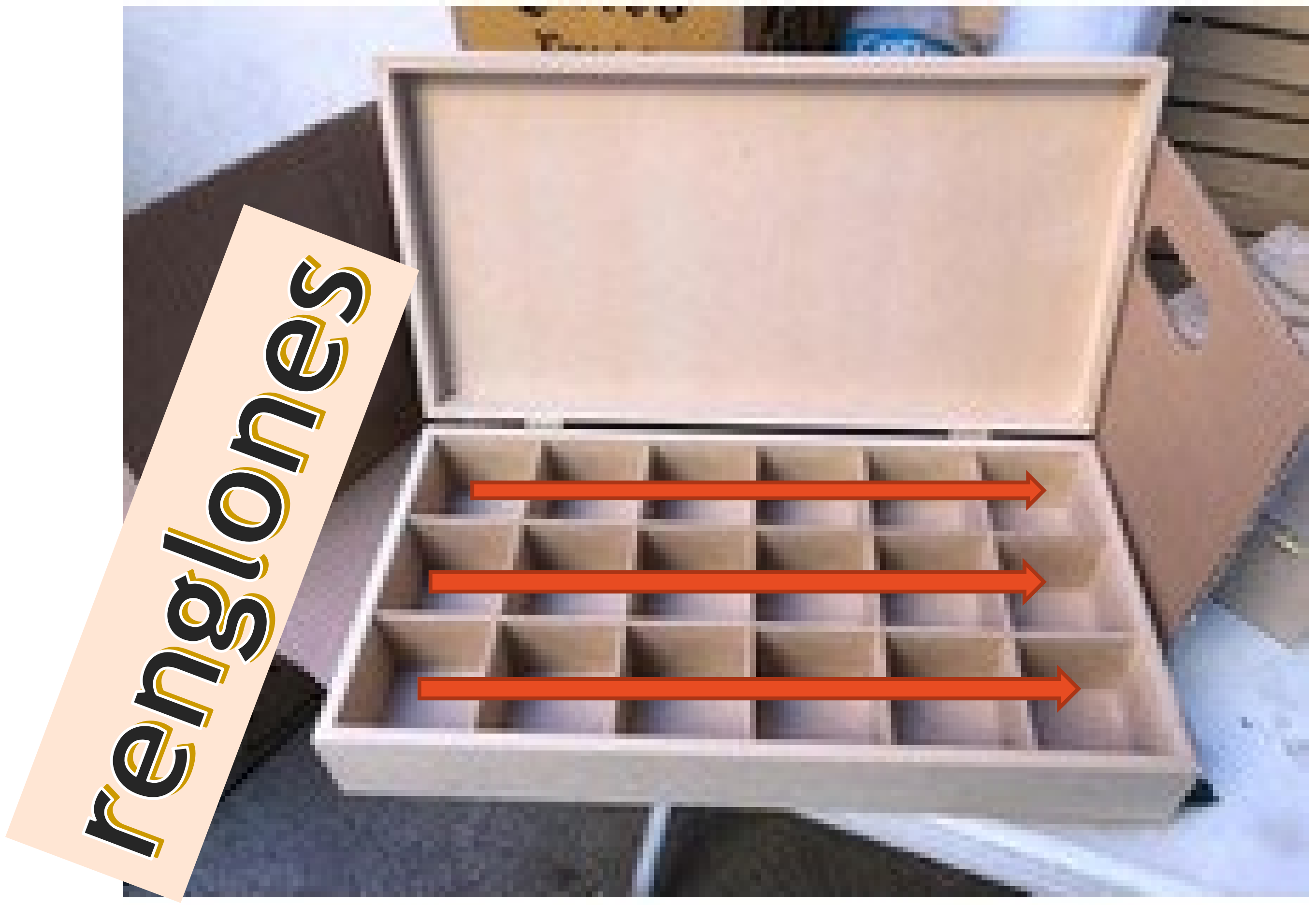
7	7	10	8

```
int resE[3] [4];
```

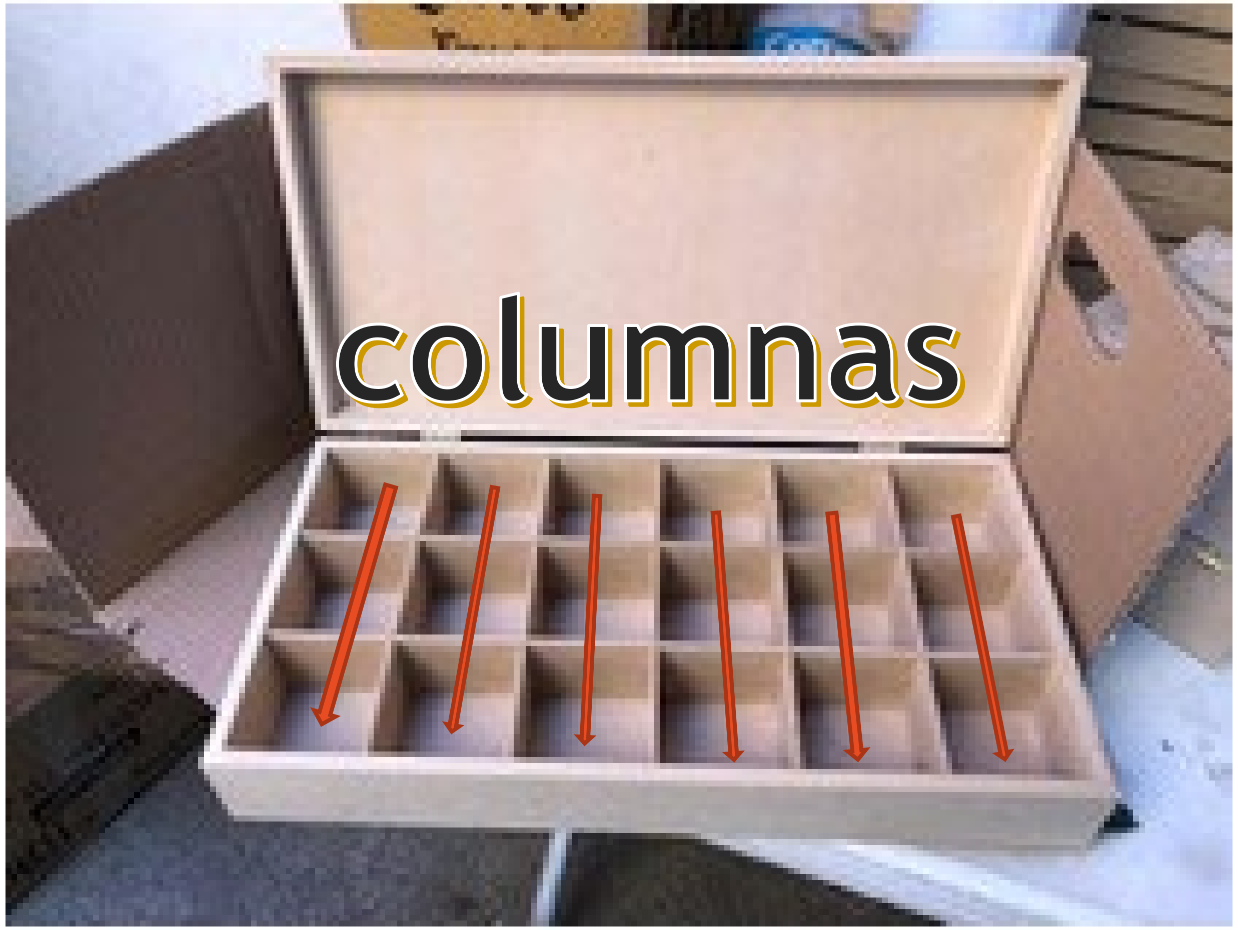
5	7	9	5
10	8	9	10
7	7	10	8

Mader Box

Arreglo bidimensional



Arreglo bidimensional



Arreglos

renglones

elementos en cada arreglo

columnas

```
int resE[3] [4];
```

5	7	9	5
10	8	9	10
7	7	10	8

matrices

Arreglo bidimensional- mismo tipo de dato

matrix

```
m <- matrix(nrow = 2, ncol = 2)
```

dim

```
dim(m)
```

```
m <- matrix(c(1:3))
```

```
class(m)
```

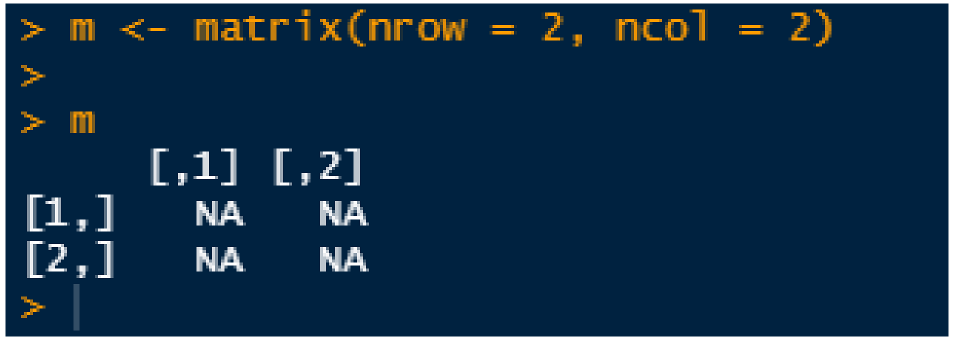
```
typeof(m)
```

matrices

Bidimentional array- same data type

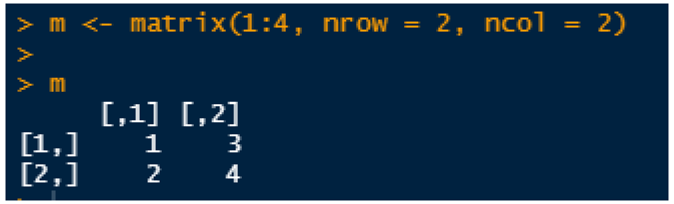
matrix

```
m <- matrix(nrow = 2, ncol = 2)
```



```
> m <- matrix(nrow = 2, ncol = 2)
>
> m
     [,1] [,2]
[1,]   NA   NA
[2,]   NA   NA
>
```

```
m <- matrix(data, nrow = 2, ncol = 2)
```

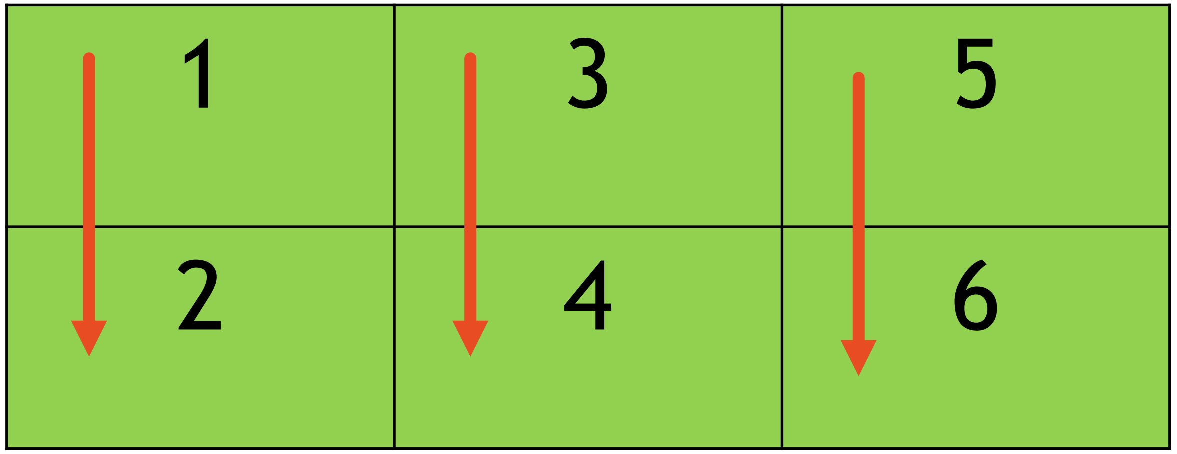


```
> m <- matrix(1:4, nrow = 2, ncol = 2)
>
> m
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```r
m <- matrix(1:6, nrow = 2, ncol = 3)
```

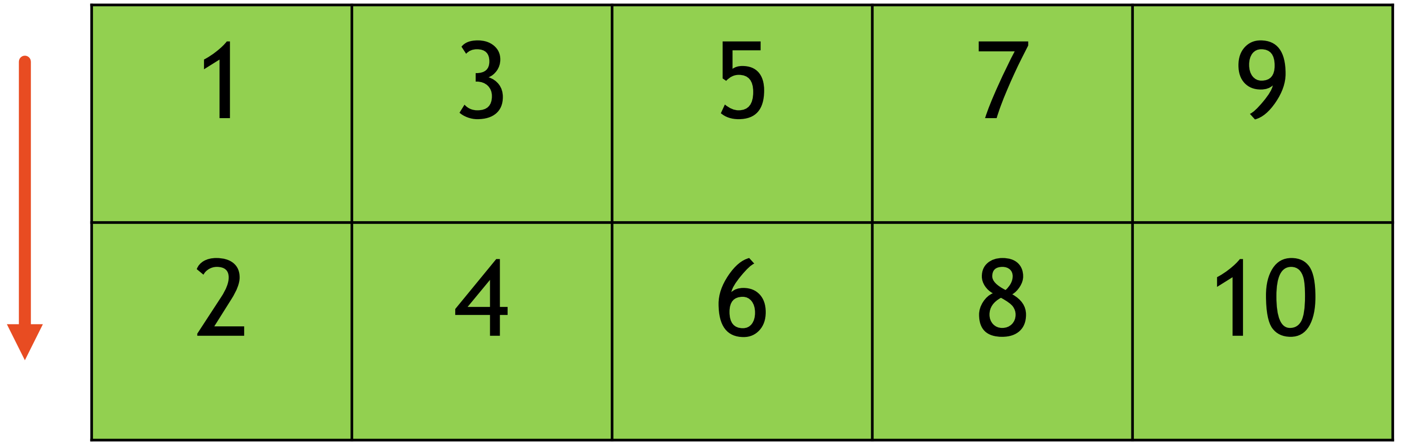
Fill by column

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



```r
m       <- 1:10
dim(m) <- c(2, 5)
```

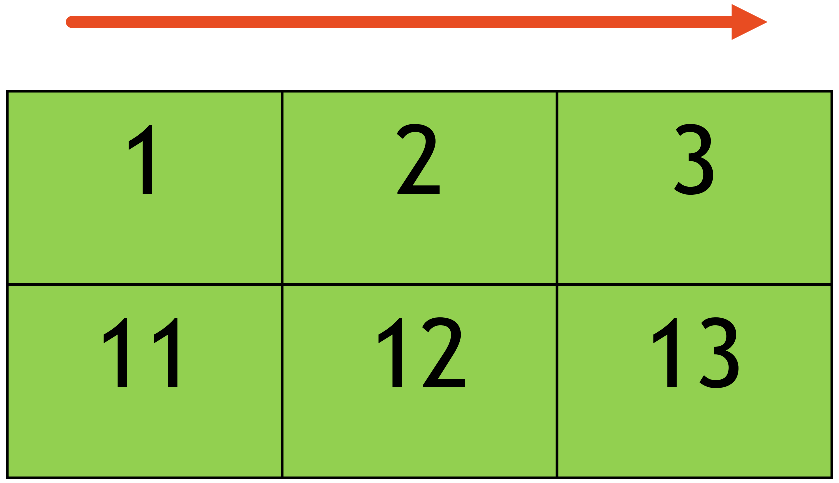
[1,2,3,4,5,6,7,8,9,10]



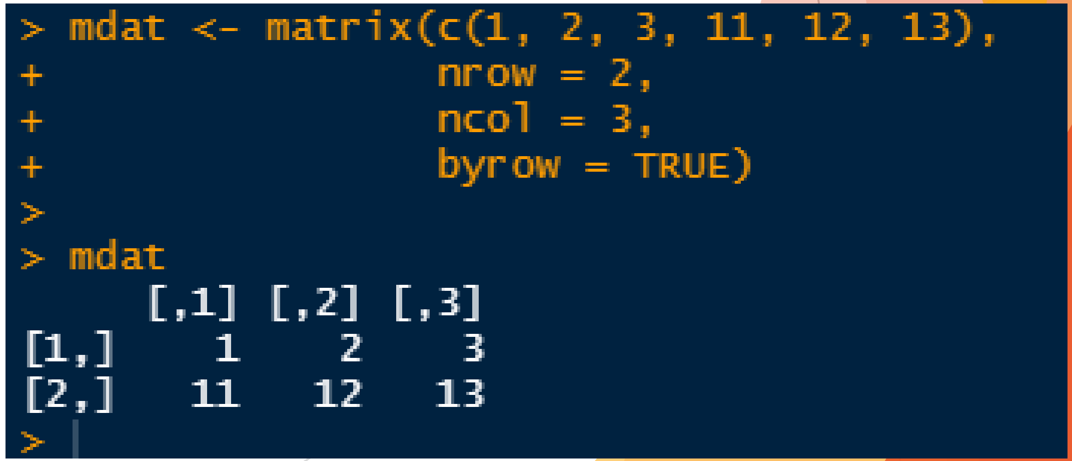
1	3	5	7	9
2	4	6	8	10

byrow

```
mdat <- matrix(c(1, 2, 3, 11, 12, 13),
               nrow = 2,
               ncol = 3,
               byrow = TRUE)
```



1	2	3
11	12	13



```
> mdat <- matrix(c(1, 2, 3, 11, 12, 13),
+                nrow = 2,
+                ncol = 3,
+                byrow = TRUE)
>
> mdat
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]   11   12   13
>
```

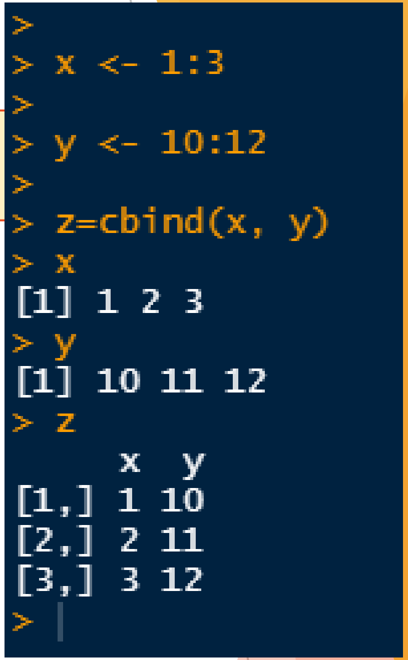
Other way to create a matrix

cbind() Combine by columns

x <- 1:3
y <- 10:12
cbind(x, y)

1	10
2	11
3	12

Vectores del mismo tamaño

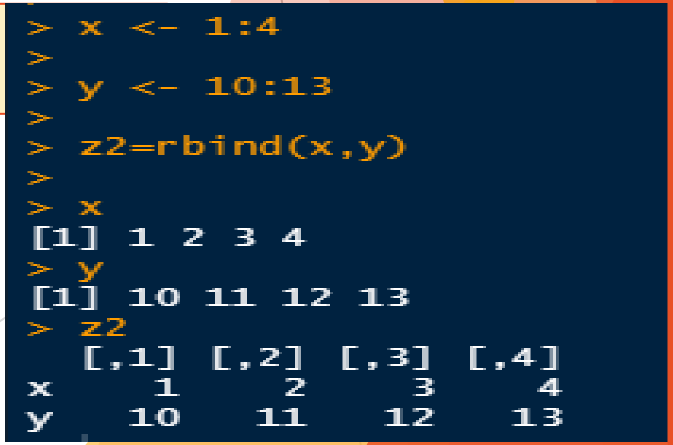


```
>
> x <- 1:3
>
> y <- 10:12
>
> z=cbind(x, y)
> x
[1] 1 2 3
> y
[1] 10 11 12
> z
     x  y
[1,] 1 10
[2,] 2 11
[3,] 3 12
>
```

rbind() Combine by rows

x <- 1:4
y <- 10:13
rbind(x, y)

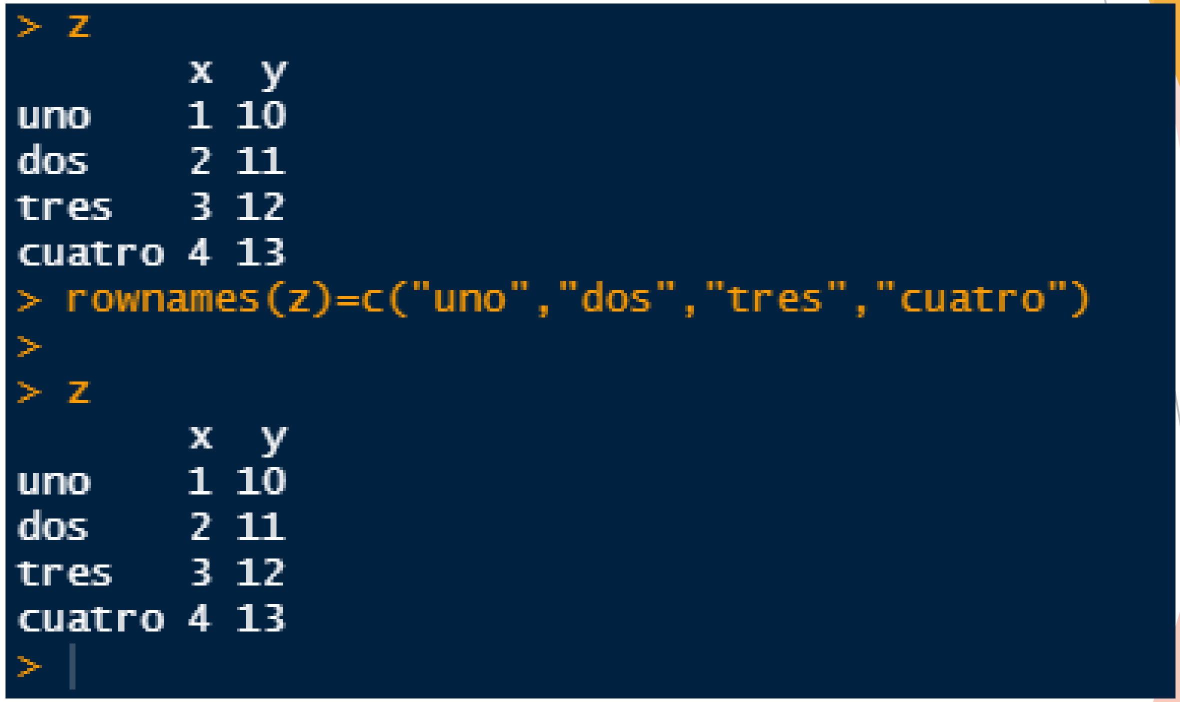
1	2	3	4
10	11	12	13



```
> x <- 1:4
>
> y <- 10:13
>
> z2=rbind(x,y)
>
> x
[1] 1 2 3 4
> y
[1] 10 11 12 13
> z2
  [,1] [,2] [,3] [,4]
x    1    2    3    4
y   10   11   12   13
```

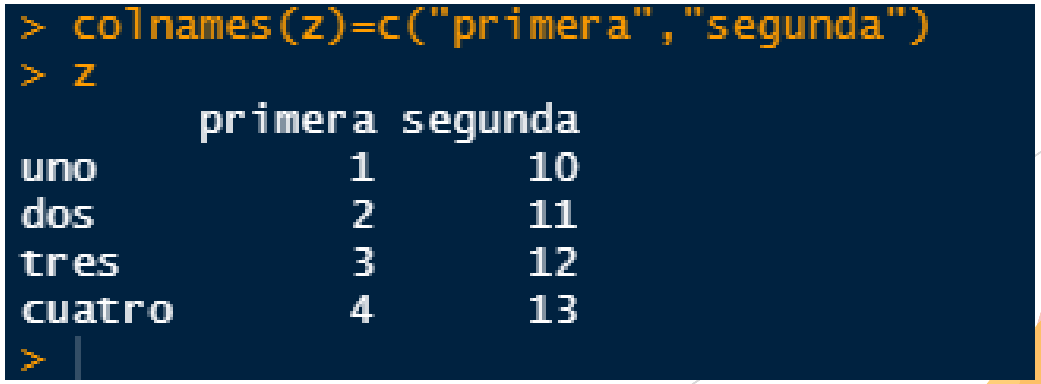
Give names to rows and cols

rownames()



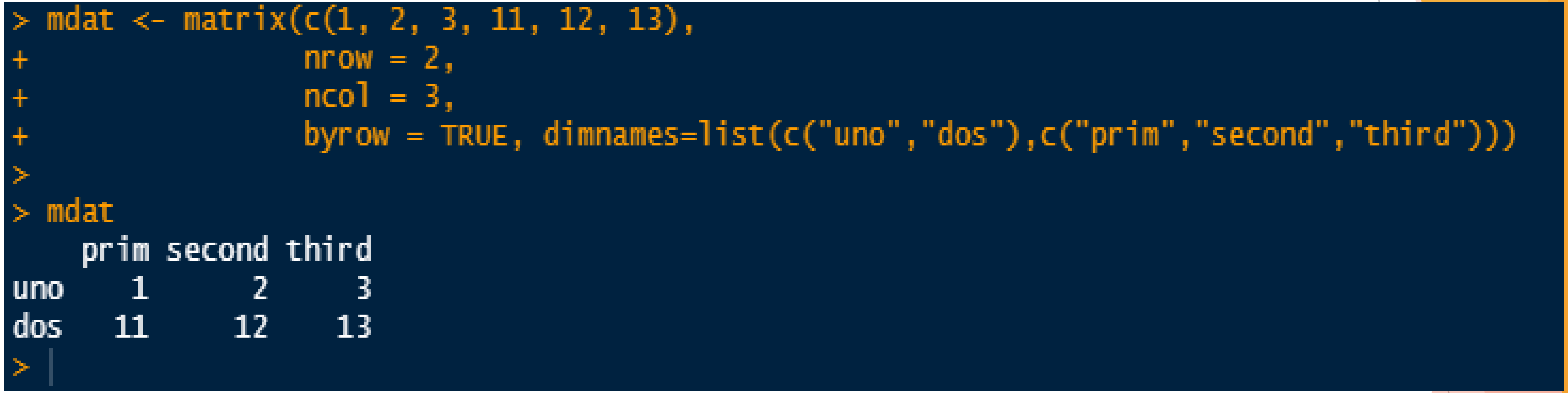
```
> z
       x  y
uno    1 10
dos    2 11
tres   3 12
cuatro 4 13
> rownames(z)=c("uno","dos","tres","cuatro")
>
> z
       x  y
uno    1 10
dos    2 11
tres   3 12
cuatro 4 13
>
```

colnames()



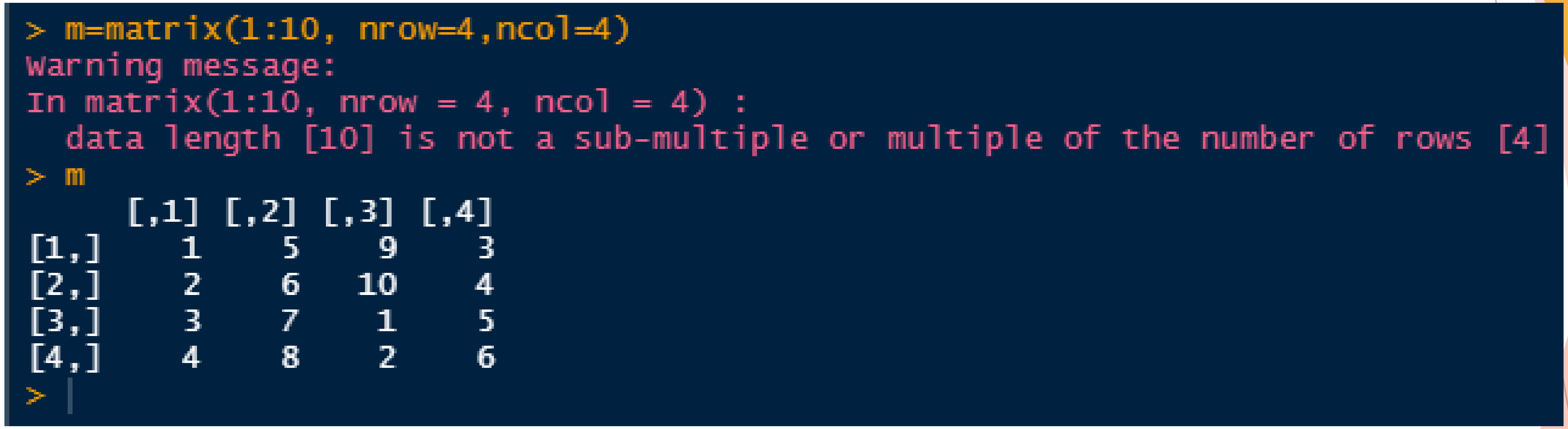
```
> colnames(z)=c("primera","segunda")
> z
       primera segunda
uno          1      10
dos          2      11
tres         3      12
cuatro       4      13
>
```

dinnames



```
> mdat <- matrix(c(1, 2, 3, 11, 12, 13),
+                nrow = 2,
+                ncol = 3,
+                byrow = TRUE, dimnames=list(c("uno","dos"),c("prim","second","third")))
>
> mdat
    prim second third
uno    1      2     3
dos   11     12    13
> 
```

recycling

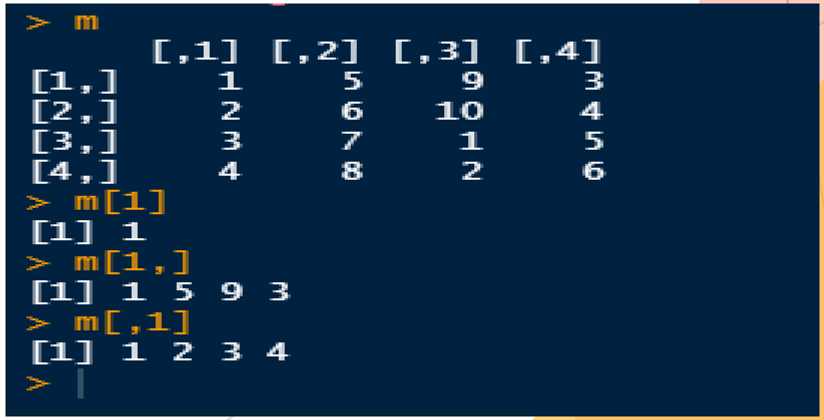


```
> m=matrix(1:10, nrow=4,ncol=4)
Warning message:
In matrix(1:10, nrow = 4, ncol = 4) :
  data length [10] is not a sub-multiple or multiple of the number of rows [4]
> m
     [,1] [,2] [,3] [,4]
[1,]    1    5    9    3
[2,]    2    6   10    4
[3,]    3    7    1    5
[4,]    4    8    2    6
> 
```

mdat[2, 3] — Accesa un elemento específico

mdat[2,] — Accesa un renglón específico

mdat[,3] — Accesa una columna específica as vector



```
> m
     [,1] [,2] [,3] [,4]
[1,]    1    5    9    3
[2,]    2    6   10    4
[3,]    3    7    1    5
[4,]    4    8    2    6
> m[1]
[1] 1
> m[1,]
[1] 1 5 9 3
> m[,1]
[1] 1 2 3 4
> 
```

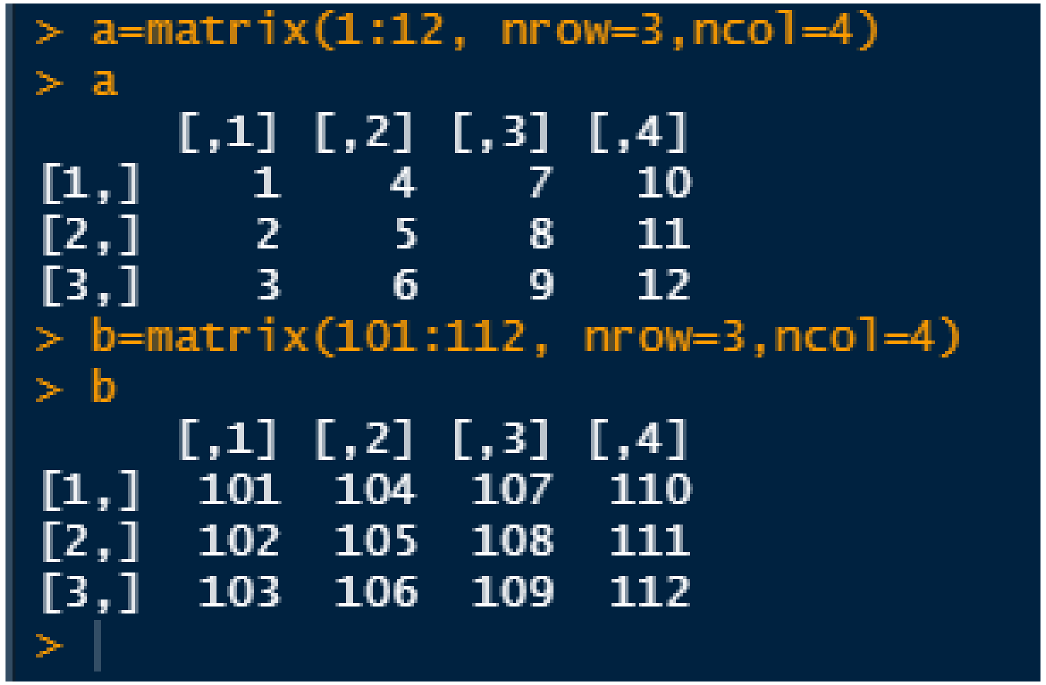
Assign names

```
dimnames(mdat) = list( c("row1", "row2"),
c("col1", "col2", "col3"))
```

Access by names

```
mdat["row2", "col3"]
```

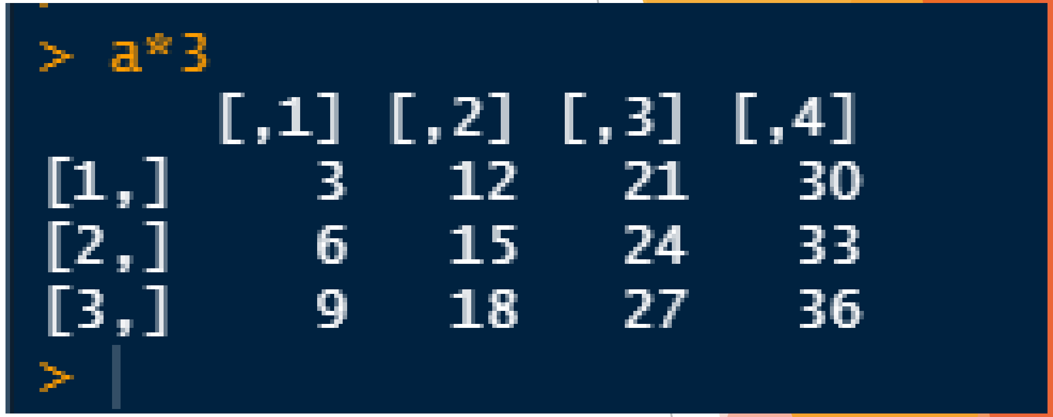
Matrix operations



```
> a=matrix(1:12, nrow=3,ncol=4)
> a
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> b=matrix(101:112, nrow=3,ncol=4)
> b
     [,1] [,2] [,3] [,4]
[1,]  101  104  107  110
[2,]  102  105  108  111
[3,]  103  106  109  112
> 
```

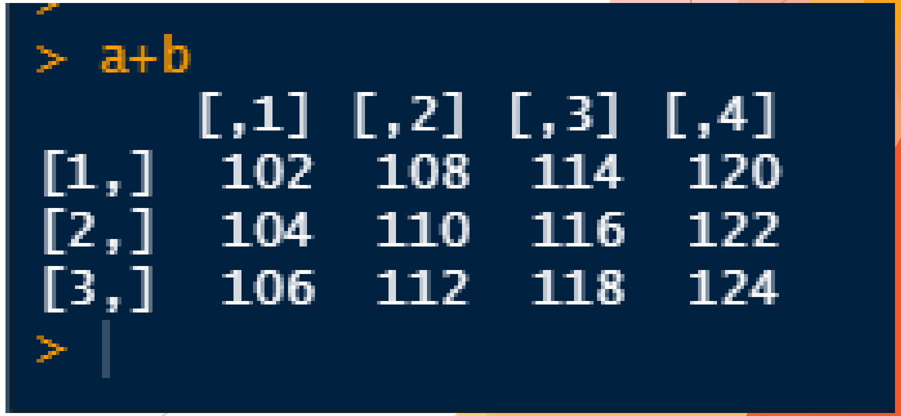
element-wise

Matrix* scalar



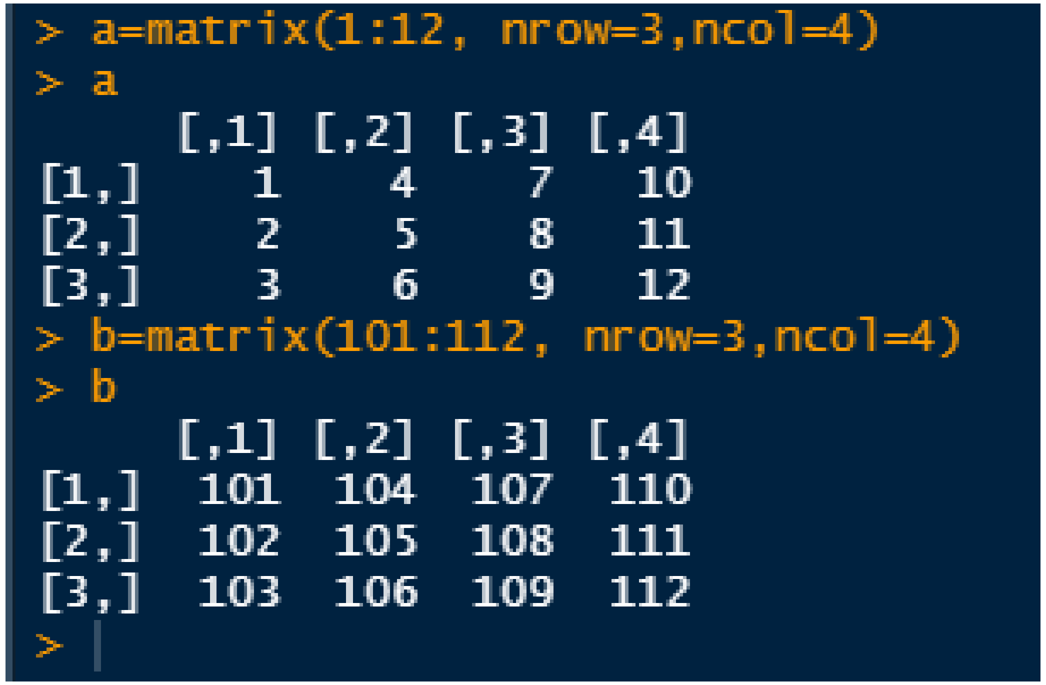
```
> a*3
     [,1] [,2] [,3] [,4]
[1,]    3   12   21   30
[2,]    6   15   24   33
[3,]    9   18   27   36
> 
```

MatrixA+MatrixB

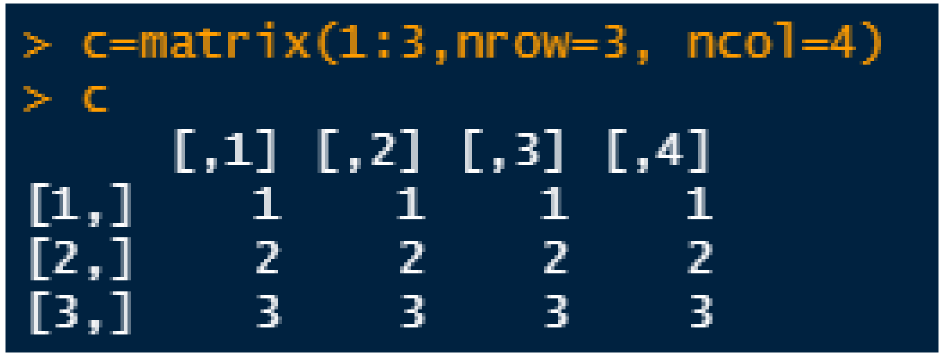


```
> a+b
     [,1] [,2] [,3] [,4]
[1,]  102  108  114  120
[2,]  104  110  116  122
[3,]  106  112  118  124
> 
```

Matrix operations

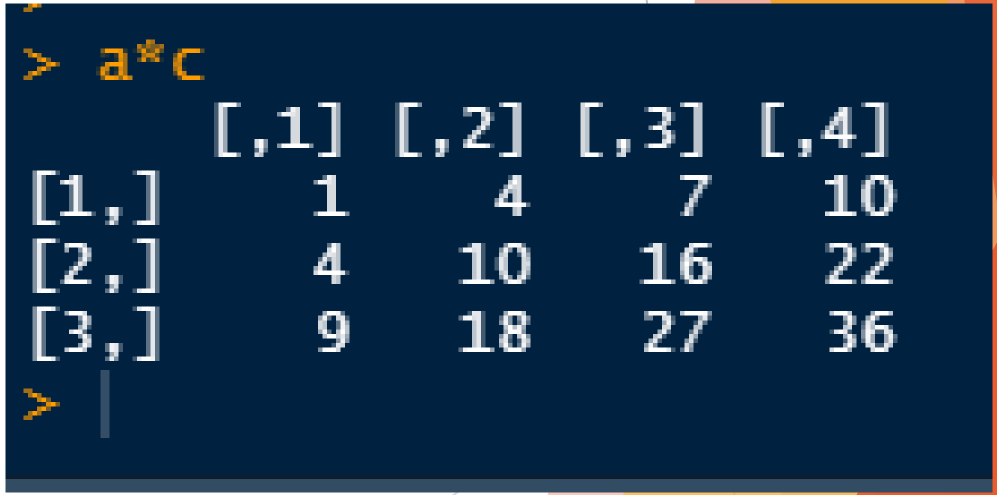


```
> a=matrix(1:12, nrow=3,ncol=4)
> a
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> b=matrix(101:112, nrow=3,ncol=4)
> b
     [,1] [,2] [,3] [,4]
[1,]  101  104  107  110
[2,]  102  105  108  111
[3,]  103  106  109  112
> 
```



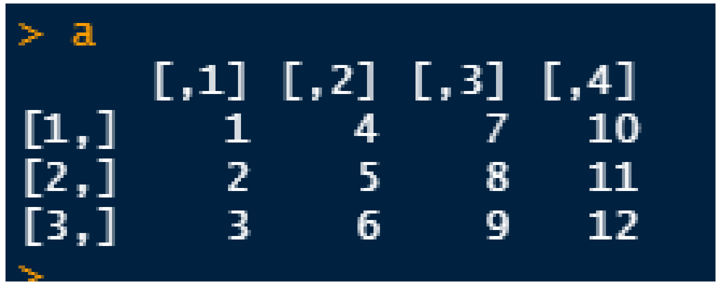
```
> c=matrix(1:3,nrow=3, ncol=4)
> c
     [,1] [,2] [,3] [,4]
[1,]    1    1    1    1
[2,]    2    2    2    2
[3,]    3    3    3    3
```

MatrixA*MatrixB



```
> a*c
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    4   10   16   22
[3,]    9   18   27   36
> 
```

Matrix operations



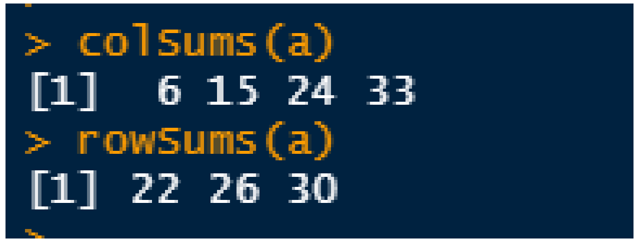
```
> a
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
>
```

colSums( )

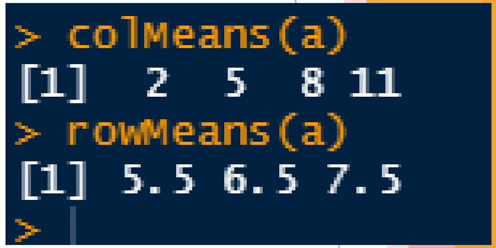
rowSums( )

colMeans( )

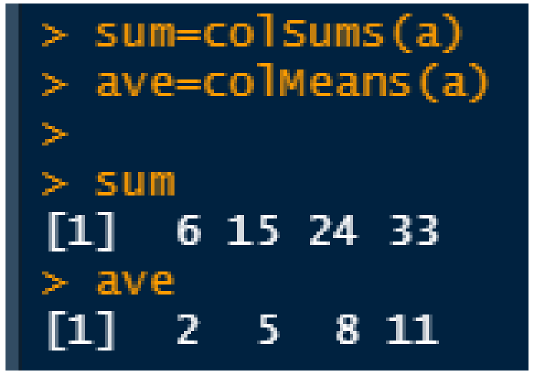
rowMeans( )



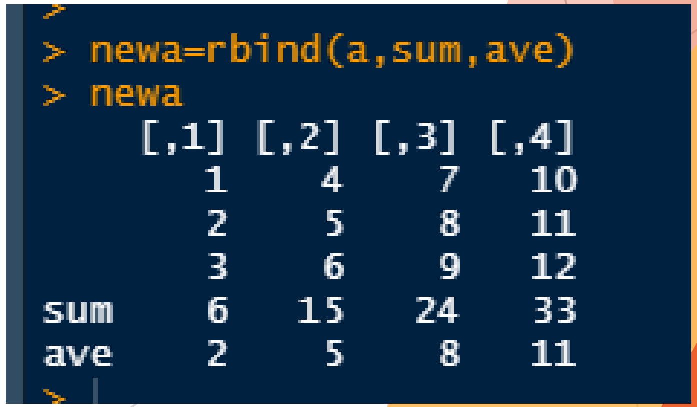
```
> colSums(a)
[1]  6 15 24 33
> rowSums(a)
[1] 22 26 30
```



```
> colMeans(a)
[1]  2  5  8 11
> rowMeans(a)
[1] 5.5 6.5 7.5
>
```



```
> sum=colSums(a)
> ave=colMeans(a)
>
> sum
[1]  6 15 24 33
> ave
[1]  2  5  8 11
```



```
>
> newa=rbind(a,sum,ave)
> newa
    [,1] [,2] [,3] [,4]
       1    4    7   10
       2    5    8   11
       3    6    9   12
sum    6   15   24   33
ave    2    5    8   11
>
```

	camisas	pantalones	Chamaras
1995	25	12	10
1996	56	45	16
1997	78	26	14
1998	98	54	9

Se pueden redimensionar

```
m      <- 1:10
dim(m) <- c(2, 5)
```

Datos perdidos

NA

No available

```
x <- c("a", NA, "c", "d", NA)
y <- c("a", "b", "c", "d", "e")
is.na(x)
anyNA(x)
```

Inf

Infinito- positivo o negativo

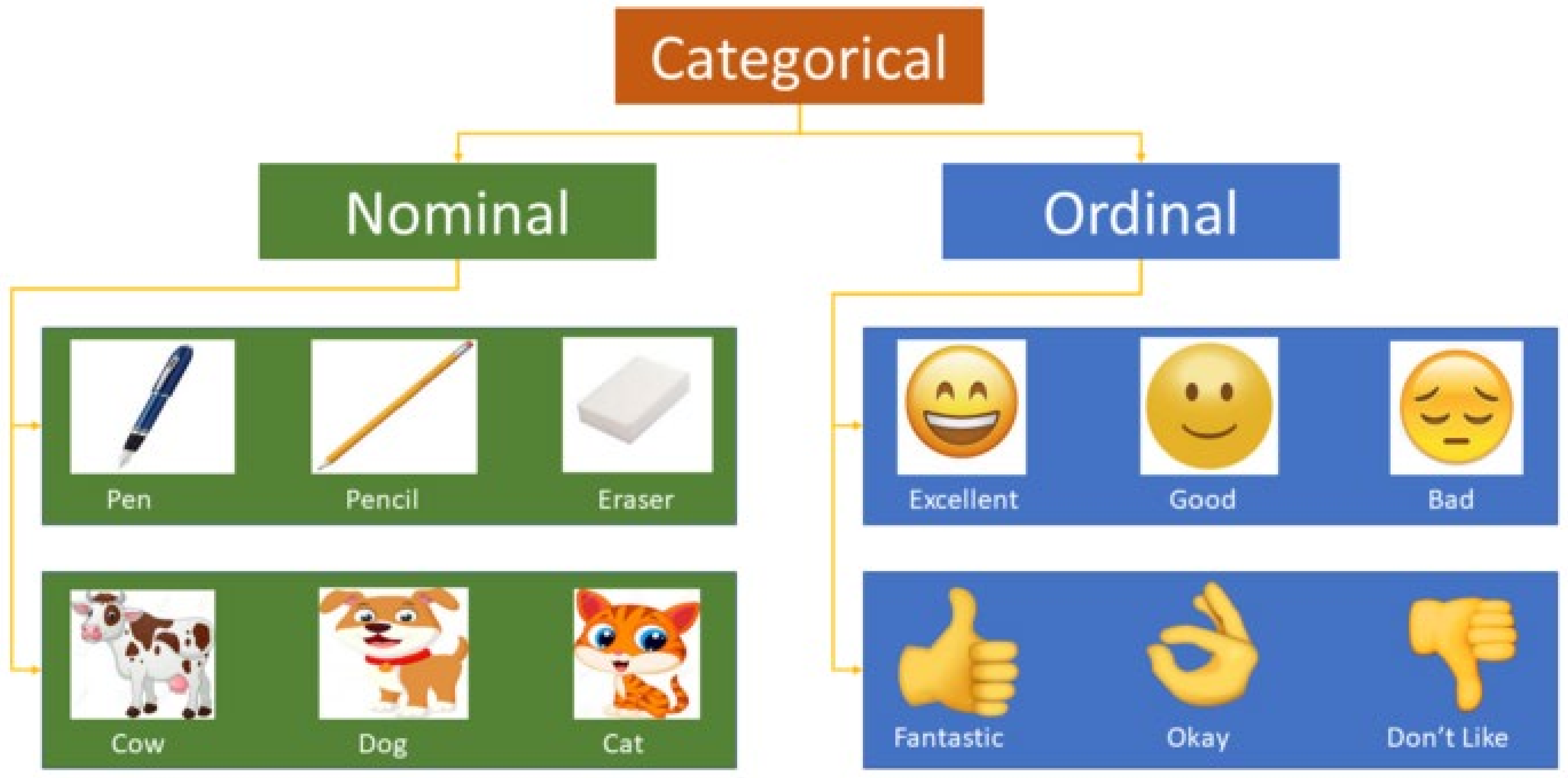
NaN

No a Number- valor indefinido – n/0

Factors

binary

✓ ✕



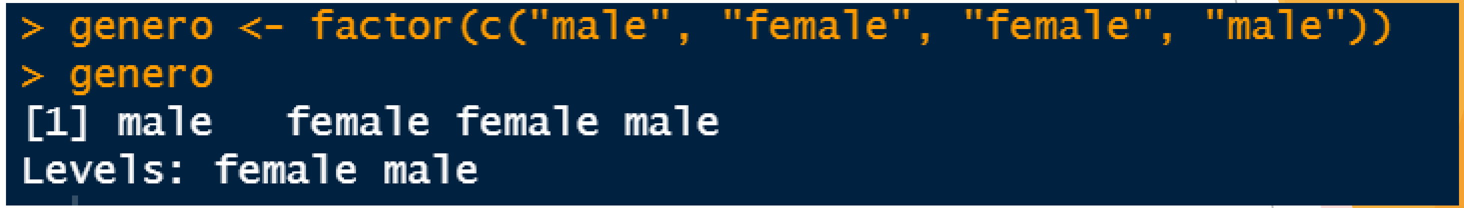
factor

- Se usan para tipo de datos categóricos
- Son almacenados como enteros
- Una vez creados solo guardan un conjunto predefinido de valores (levels)
- Los niveles se ordenan alfabéticamente
- Pueden ser ordenados o no-ordenados

factor

factor

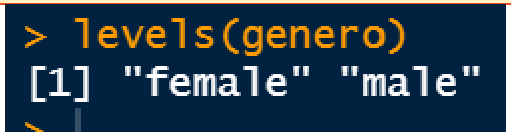
genero <- factor(c("male", "female", "female", "male"))



```
> genero <- factor(c("male", "female", "female", "male"))
> genero
[1] male   female female male
Levels: female male
```

levels

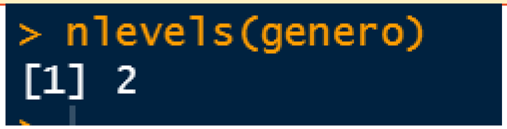
levels(genero)



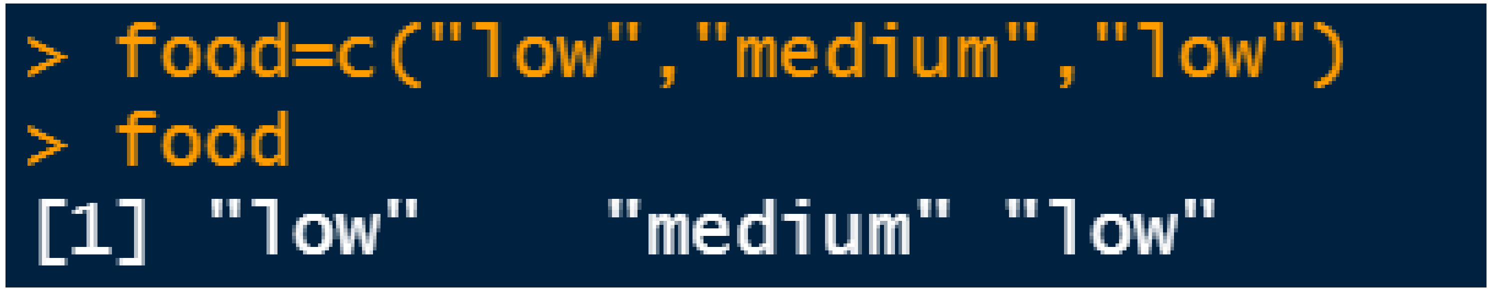
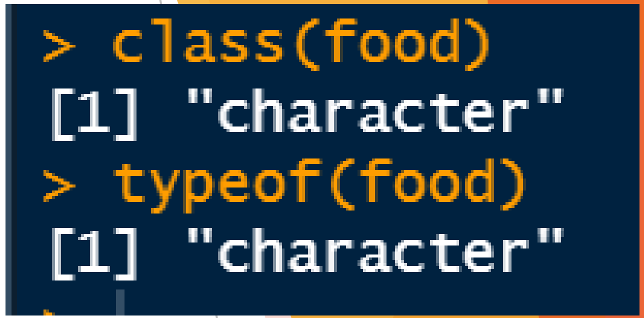
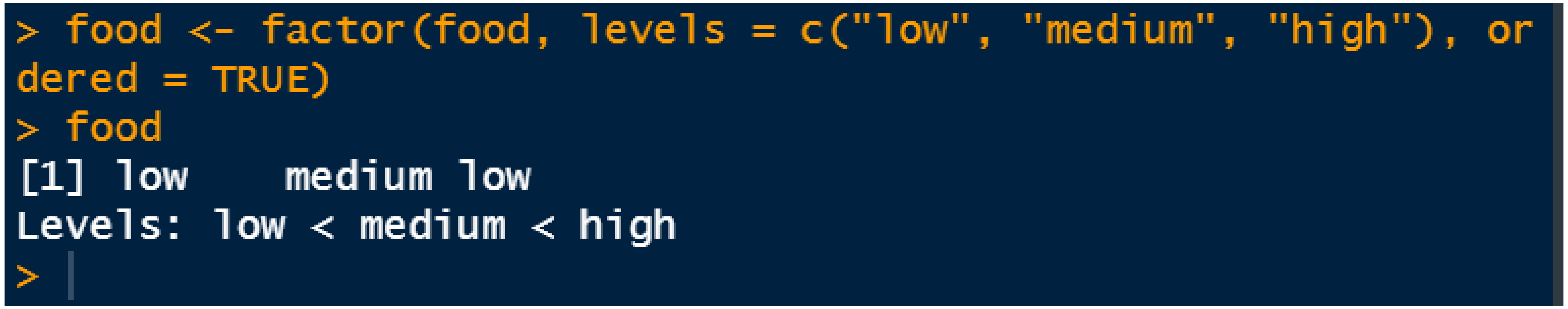
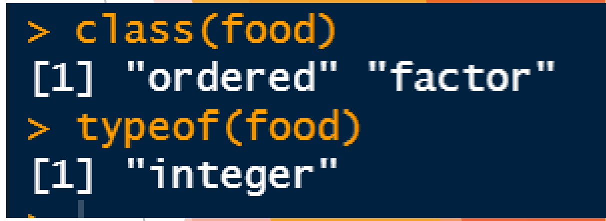
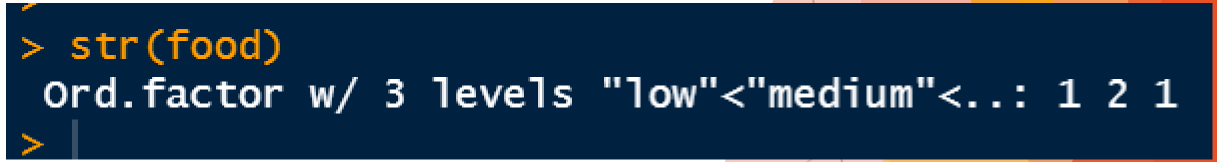
```
> levels(genero)
[1] "female" "male"
```

nlevels

nlevels(genero)



```
> nlevels(genero)
[1] 2
```

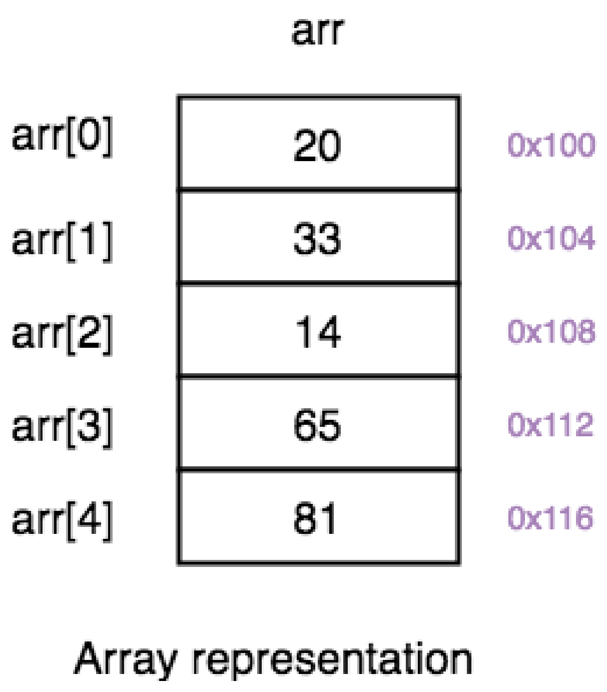

```
> food=c("low","medium","low")
> food
[1] "low"    "medium" "low"
```

```
> class(food)
[1] "character"
> typeof(food)
[1] "character"
```

```
> food <- factor(food, levels = c("low", "medium", "high"), or
dered = TRUE)
> food
[1] low    medium low
Levels: low < medium < high
> 
```

```
> class(food)
[1] "ordered" "factor"
> typeof(food)
[1] "integer"
```

```
> str(food)
 Ord.factor w/ 3 levels "low"<"medium"<..: 1 2 1
> 
```
food <- factor(food, levels = c("low", "medium", "high"), ordered = TRUE)

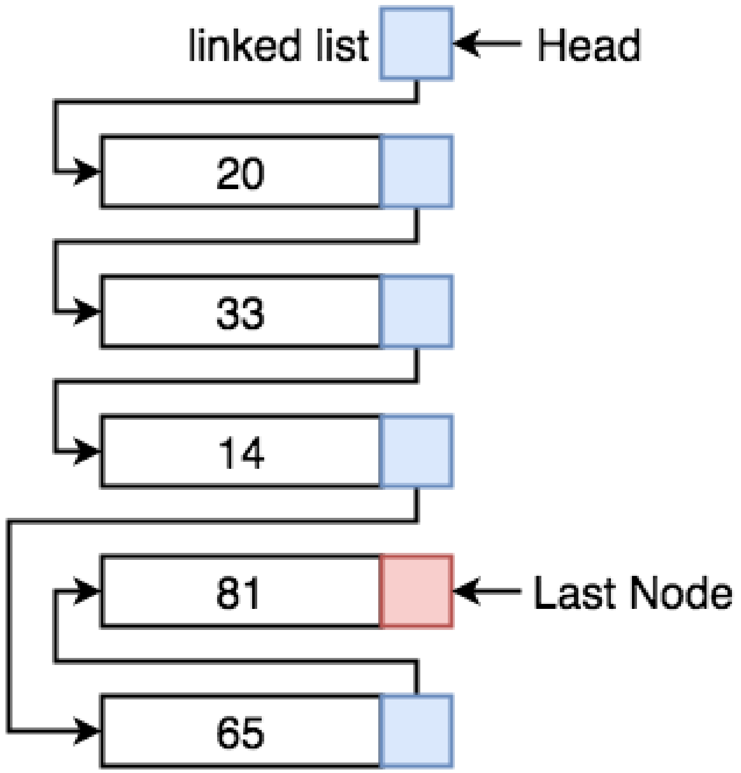
levels(food)

min(food)

Trabajan como enum

listas


arr
arr[0]
20
0x100
arr[1]
33
0x104
arr[2]
14
0x108
arr[3]
65
0x112
arr[4]
81
0x116
Array representation

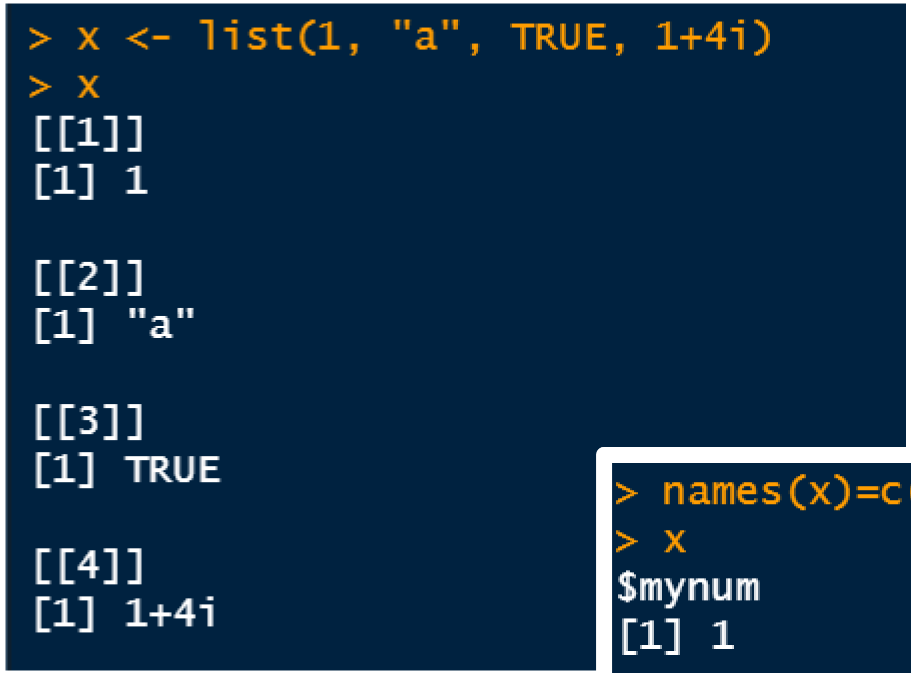

linked list
Head
20
33
14
81
Last Node
65

listas

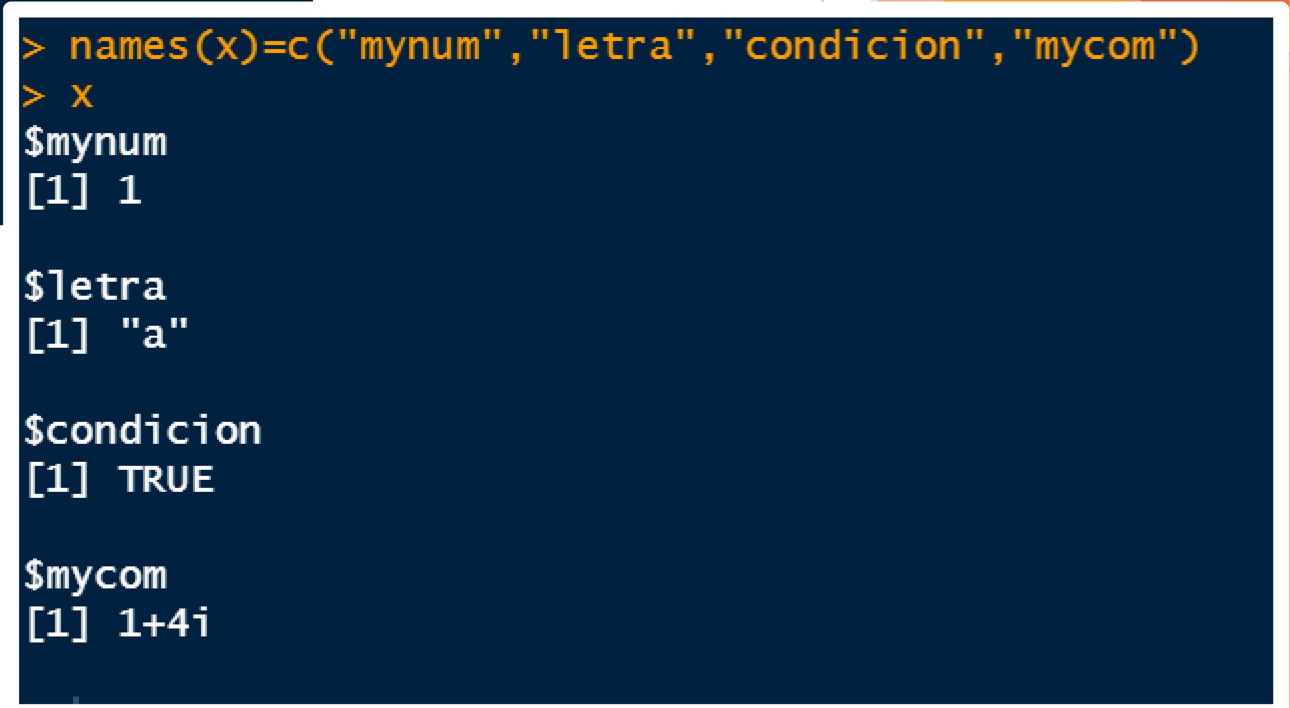
- Pueden guardar elementos de diferentes tipos
- Se les conoce como vectores genéricos
- Pueden contener otras listas
- Sus elementos pueden recibir un nombre

list

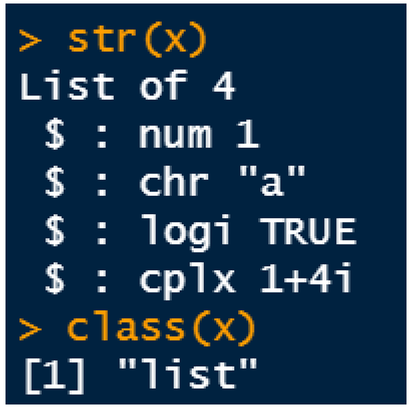
x <- list(1, "a", TRUE, 1+4i)



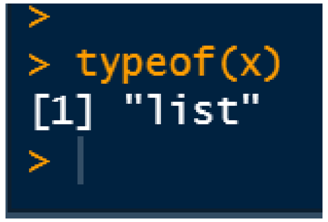
```
> x <- list(1, "a", TRUE, 1+4i)
> x
[[1]]
[1] 1

[[2]]
[1] "a"

[[3]]
[1] TRUE

[[4]]
[1] 1+4i
```



```
> names(x)=c("mynum","letra","condicion","mycom")
> x
$mynum
[1] 1

$letra
[1] "a"

$condicion
[1] TRUE

$mycom
[1] 1+4i
```

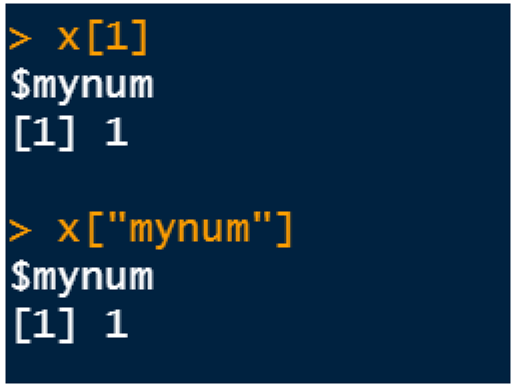


```
> str(x)
List of 4
 $ : num 1
 $ : chr "a"
 $ : logi TRUE
 $ : cplx 1+4i
> class(x)
[1] "list"
```

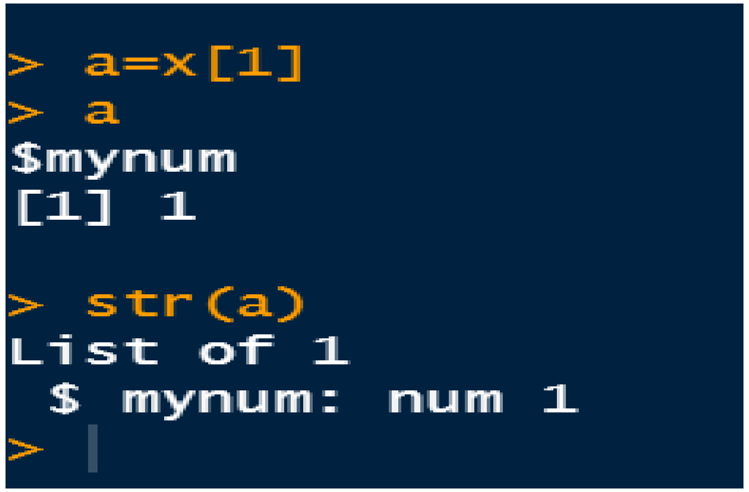


```
>
> typeof(x)
[1] "list"
> 
```

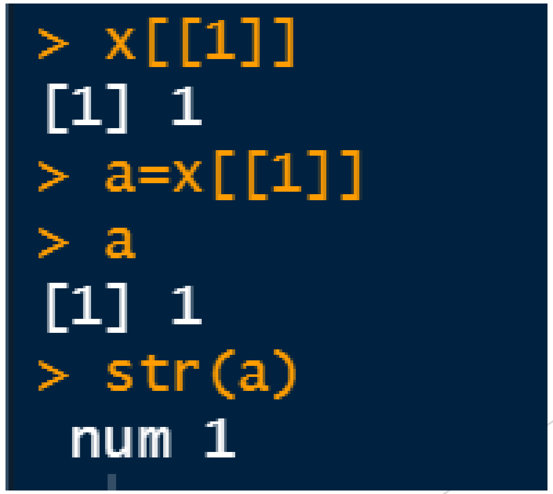
Accessing elements



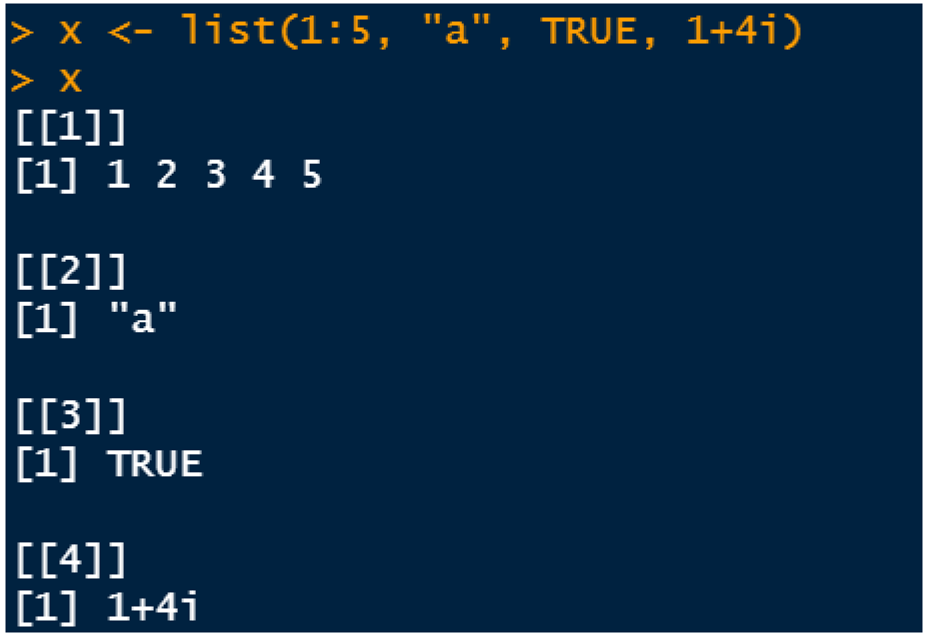
```
> x[1]
$mynum
[1] 1

> x["mynum"]
$mynum
[1] 1
```



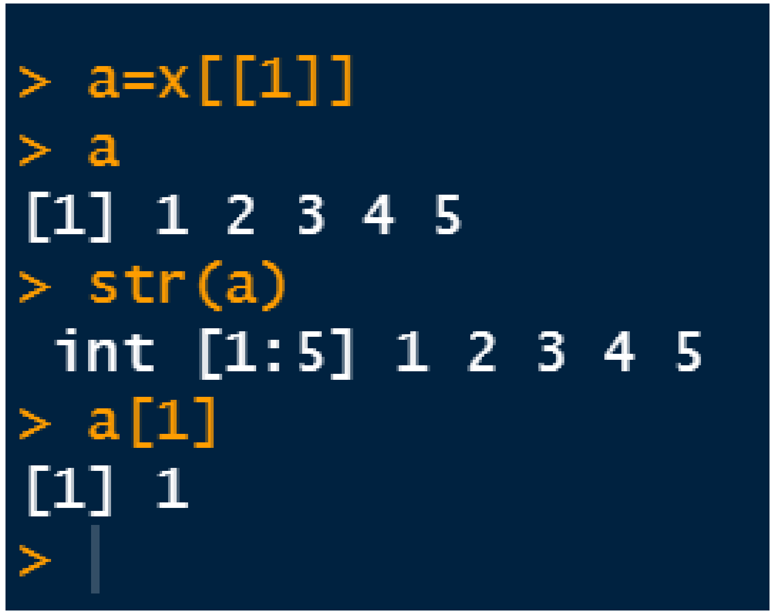
```
> a=x[1]
> a
$mynum
[1] 1

> str(a)
List of 1
 $ mynum: num 1
>
```



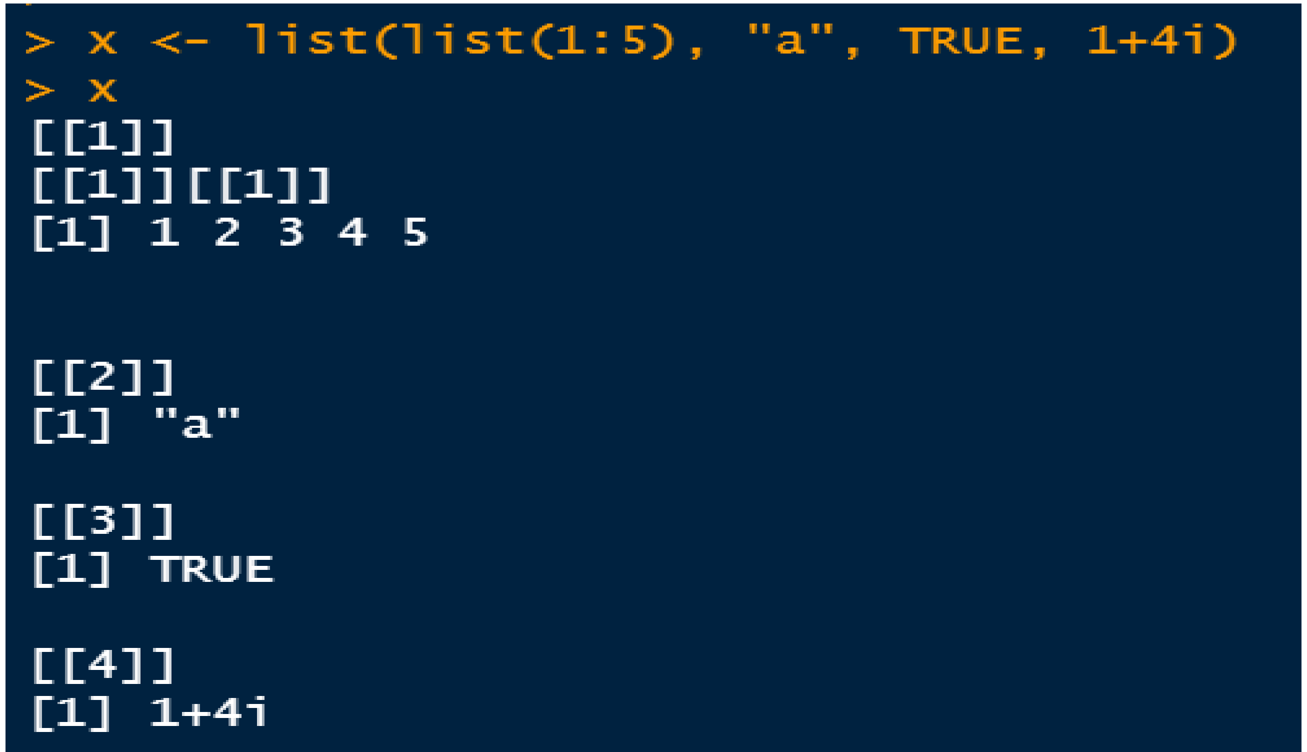
```
> x[[1]]
[1] 1
> a=x[[1]]
> a
[1] 1
> str(a)
 num 1
```



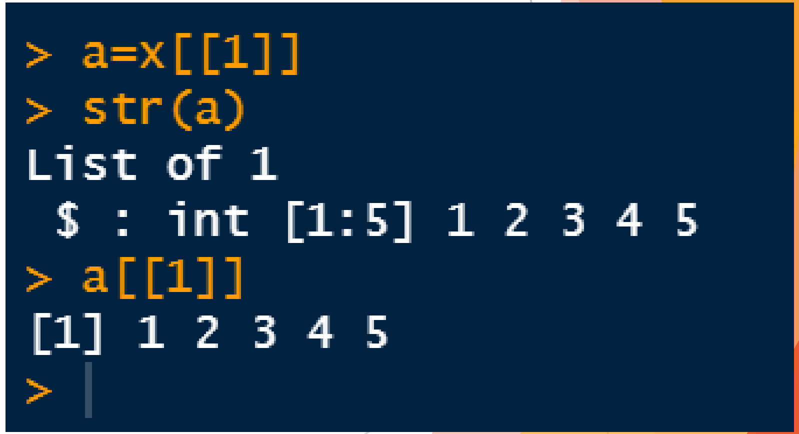
```
> x <- list(1:5, "a", TRUE, 1+4i)
> x
[[1]]
[1] 1 2 3 4 5

[[2]]
[1] "a"

[[3]]
[1] TRUE

[[4]]
[1] 1+4i
```



```
> a=x[[1]]
> a
[1] 1 2 3 4 5
> str(a)
 int [1:5] 1 2 3 4 5
> a[1]
[1] 1
>
```



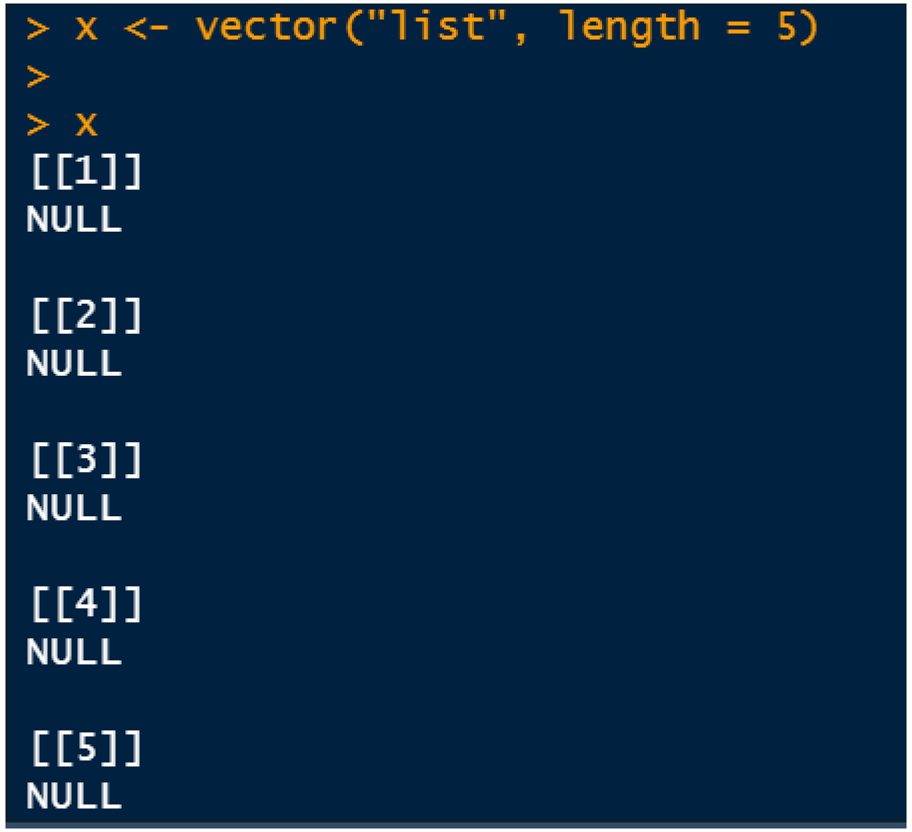
```
> x <- list(list(1:5), "a", TRUE, 1+4i)
> x
[[1]]
[[1]][[1]]
[1] 1 2 3 4 5


[[2]]
[1] "a"

[[3]]
[1] TRUE

[[4]]
[1] 1+4i
```



```
> a=x[[1]]
> str(a)
List of 1
 $ : int [1:5] 1 2 3 4 5
> a[[1]]
[1] 1 2 3 4 5
>
```

vector

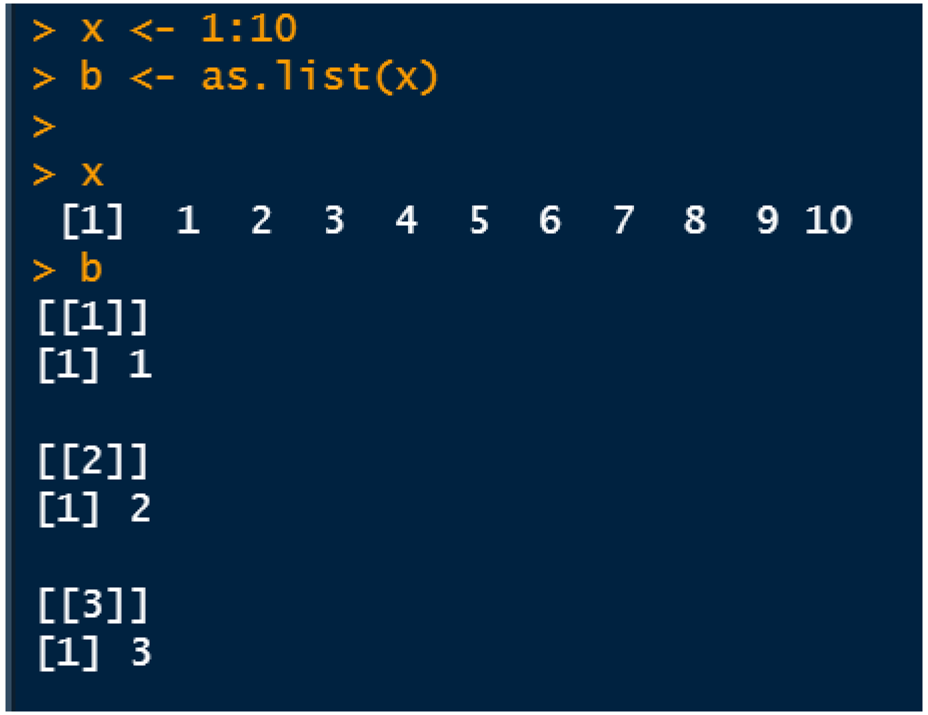
```
x <- vector("list", length = 5)
```



```
> x <- vector("list", length = 5)
>
> x
[[1]]
NULL

[[2]]
NULL

[[3]]
NULL

[[4]]
NULL

[[5]]
NULL
```

as.list

```
x <- 1:10
x <- as.list(x)
```



```
> x <- 1:10
> b <- as.list(x)
>
> x
 [1]  1  2  3  4  5  6  7  8  9 10
> b
[[1]]
[1] 1

[[2]]
[1] 2

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

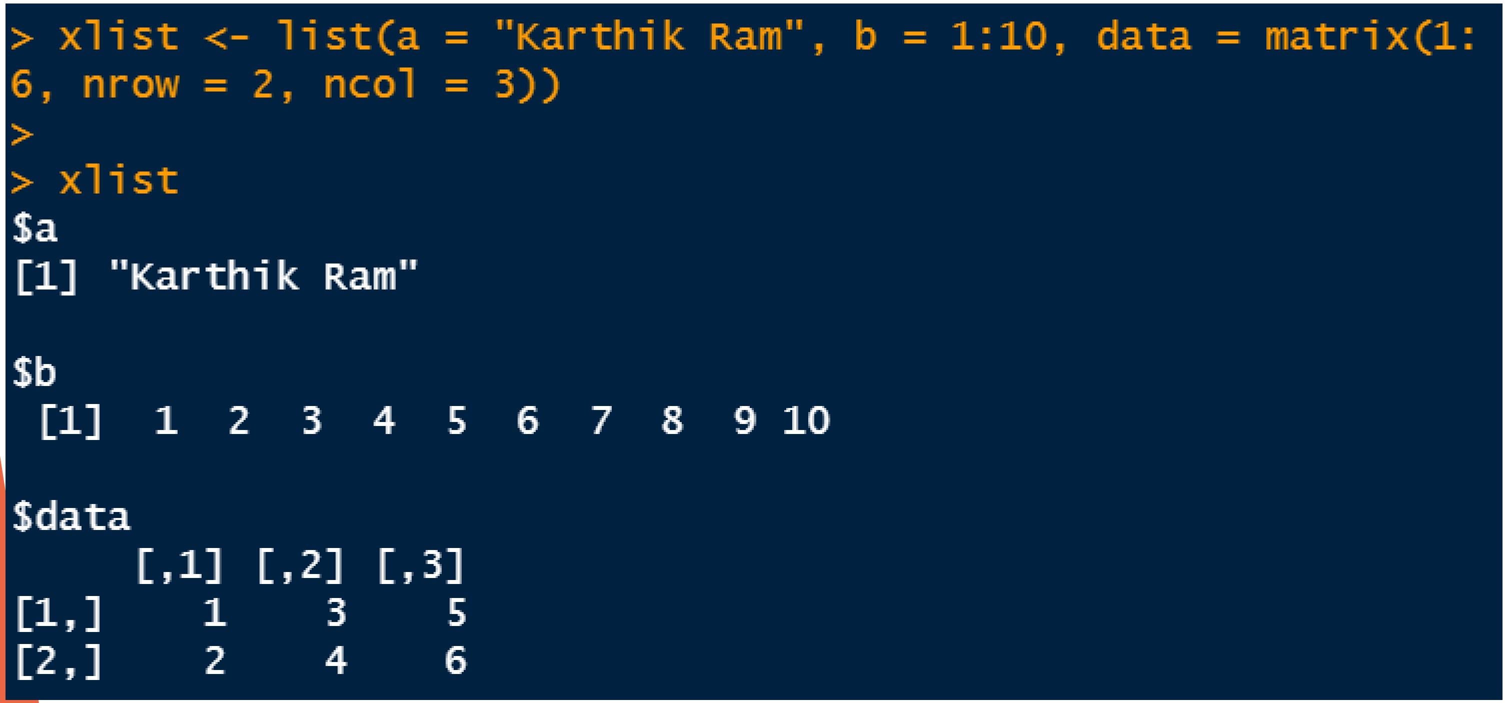
xlist <- list(a = "Karthik Ram", b = 1:10, data = matrix(1:6, nrow = 2, ncol = 3))

names

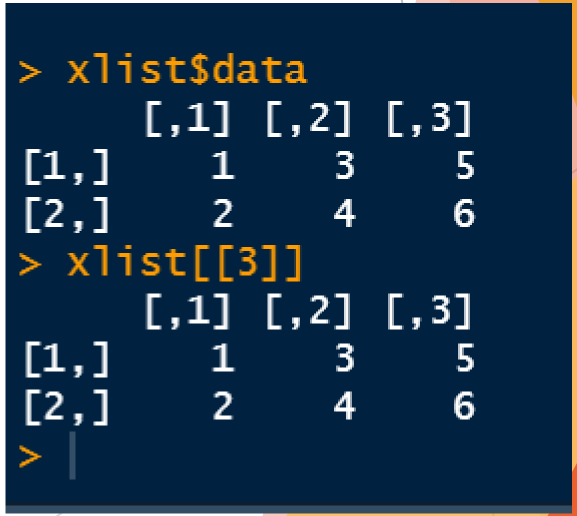
names(xlist)

$

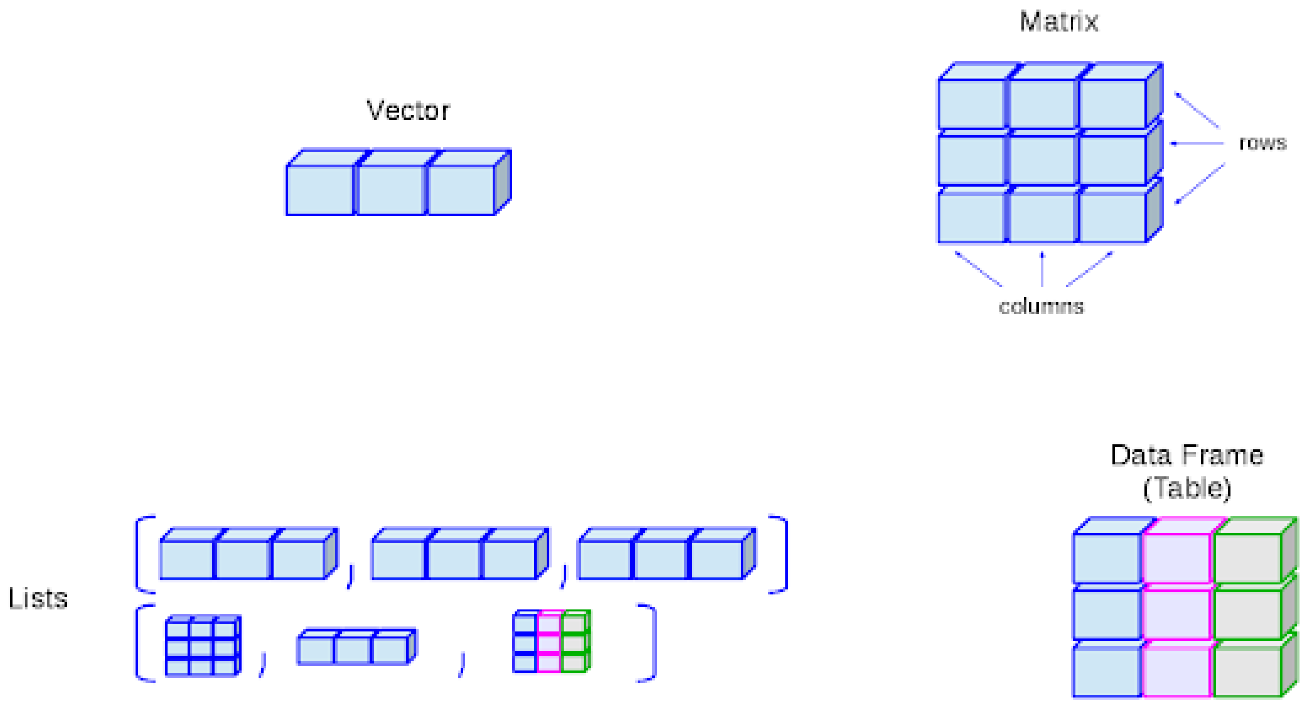
xlist$data



```
> xlist <- list(a = "Karthik Ram", b = 1:10, data = matrix(1:
6, nrow = 2, ncol = 3))
>
> xlist
$a
[1] "Karthik Ram"

$b
 [1]  1  2  3  4  5  6  7  8  9 10

$data
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```



```
> xlist$data
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> xlist[[3]]
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
> 
```



Data Frame = lista de listas

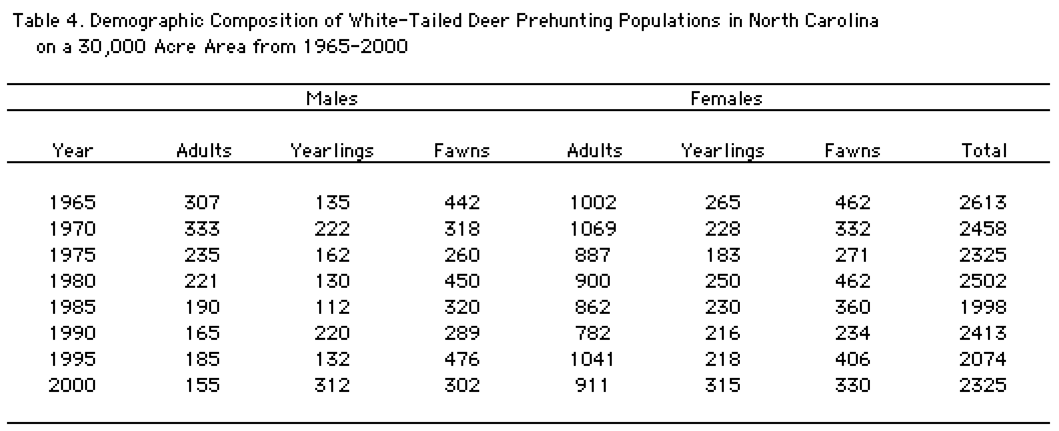
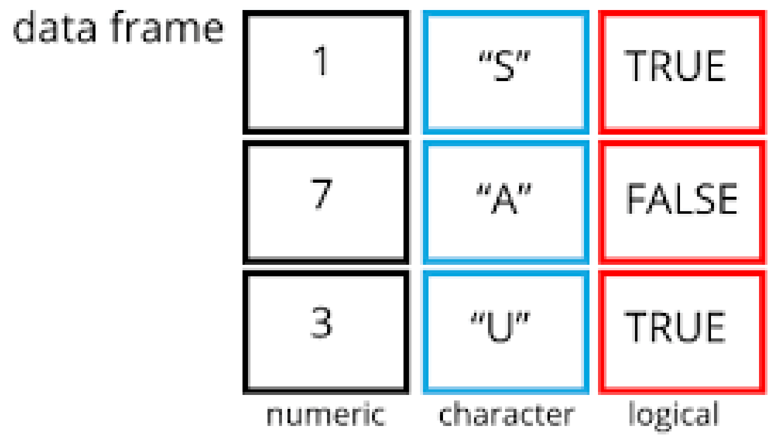
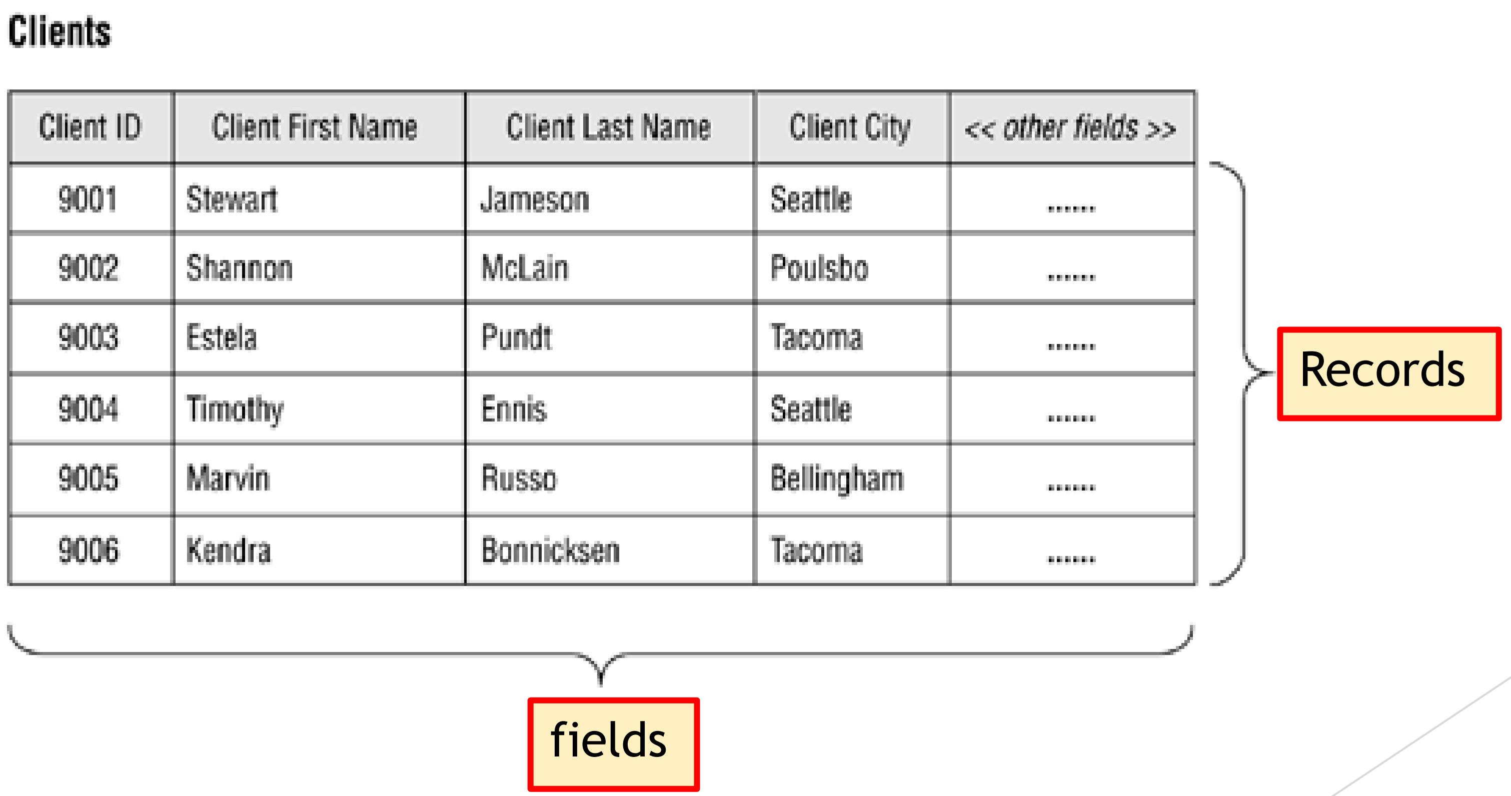


Table 4. Demographic Composition of White-Tailed Deer Prehunting Populations in North Carolina on a 30,000 Acre Area from 1965-2000

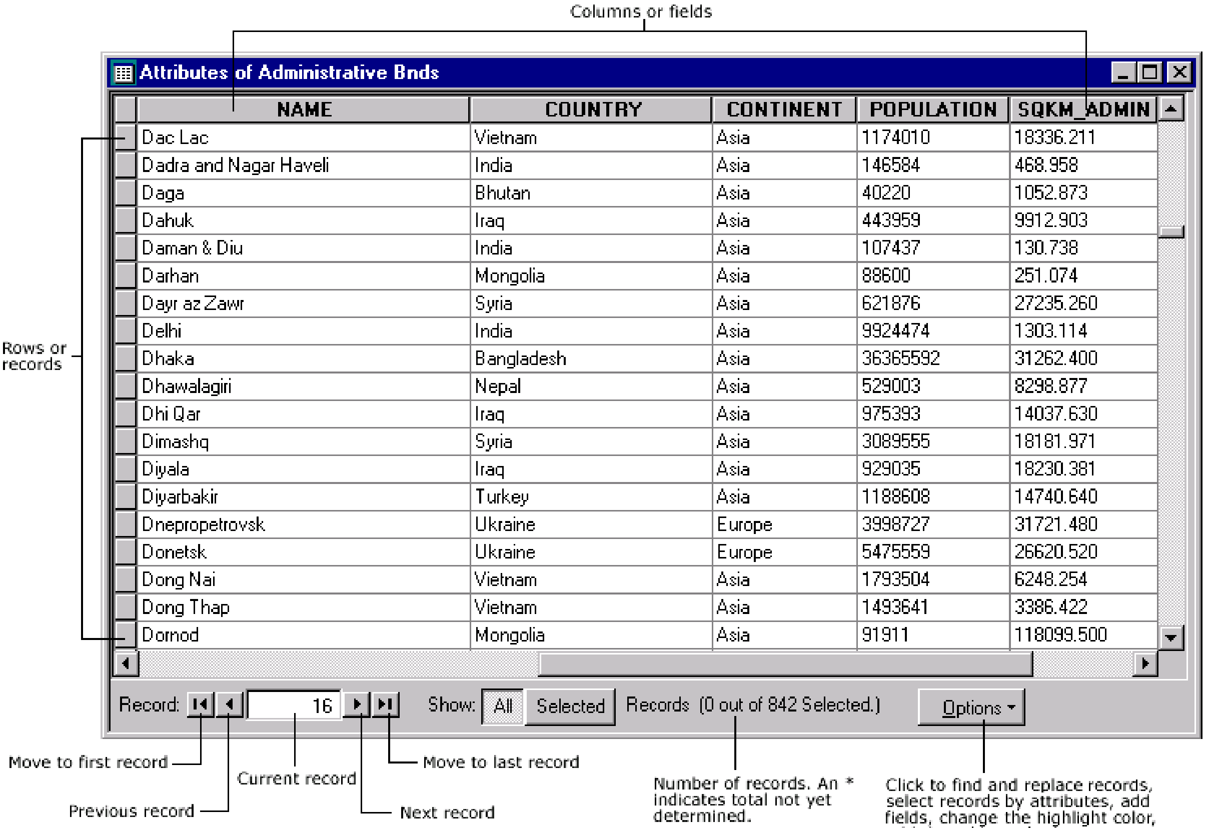
	Males			Females			
Year	Adults	Yearlings	Fawns	Adults	Yearlings	Fawns	Total
1965	307	135	442	1002	265	462	2613
1970	333	222	318	1069	228	332	2458
1975	235	162	260	887	183	271	2325
1980	221	130	450	900	250	462	2502
1985	190	112	320	862	230	360	1998
1990	165	220	289	782	216	234	2413
1995	185	132	476	1041	218	406	2074
2000	155	312	302	911	315	330	2325





Clients

Client ID	Client First Name	Client Last Name	Client City	*<< other fields >>*
9001	Stewart	Jameson	Seattle	......
9002	Shannon	McLain	Poulsbo	......
9003	Estela	Pundt	Tacoma	......
9004	Timothy	Ennis	Seattle	......
9005	Marvin	Russo	Bellingham	......
9006	Kendra	Bonnicksen	Tacoma	......



Attributes of Administrative Bnds

NAME	COUNTRY	CONTINENT	POPULATION	SQKM_ADMIN
Dac Lac	Vietnam	Asia	1174010	18336.211
Dadra and Nagar Haveli	India	Asia	146584	468.958
Daga	Bhutan	Asia	40220	1052.873
Dahuk	Iraq	Asia	443959	9912.903
Daman & Diu	India	Asia	107437	130.738
Darhan	Mongolia	Asia	88600	251.074
Dayr az Zawr	Syria	Asia	621876	27235.260
Delhi	India	Asia	9924474	1303.114
Dhaka	Bangladesh	Asia	36365592	31262.400
Dhawalagiri	Nepal	Asia	529003	8298.877
Dhi Qar	Iraq	Asia	975393	14037.630
Dimashq	Syria	Asia	3089555	18181.971
Diyala	Iraq	Asia	929035	18230.381
Diyarbakir	Turkey	Asia	1188608	14740.640
Dnepropetrovsk	Ukraine	Europe	3998727	31721.480
Donetsk	Ukraine	Europe	5475559	26620.520
Dong Nai	Vietnam	Asia	1793504	6248.254
Dong Thap	Vietnam	Asia	1493641	3386.422
Dornod	Mongolia	Asia	91911	118099.500

Record: 16 Show: All Selected Records (0 out of 842 Selected.) Options

Columns or fields

Rows or records

Move to first record

Current record

Previous record

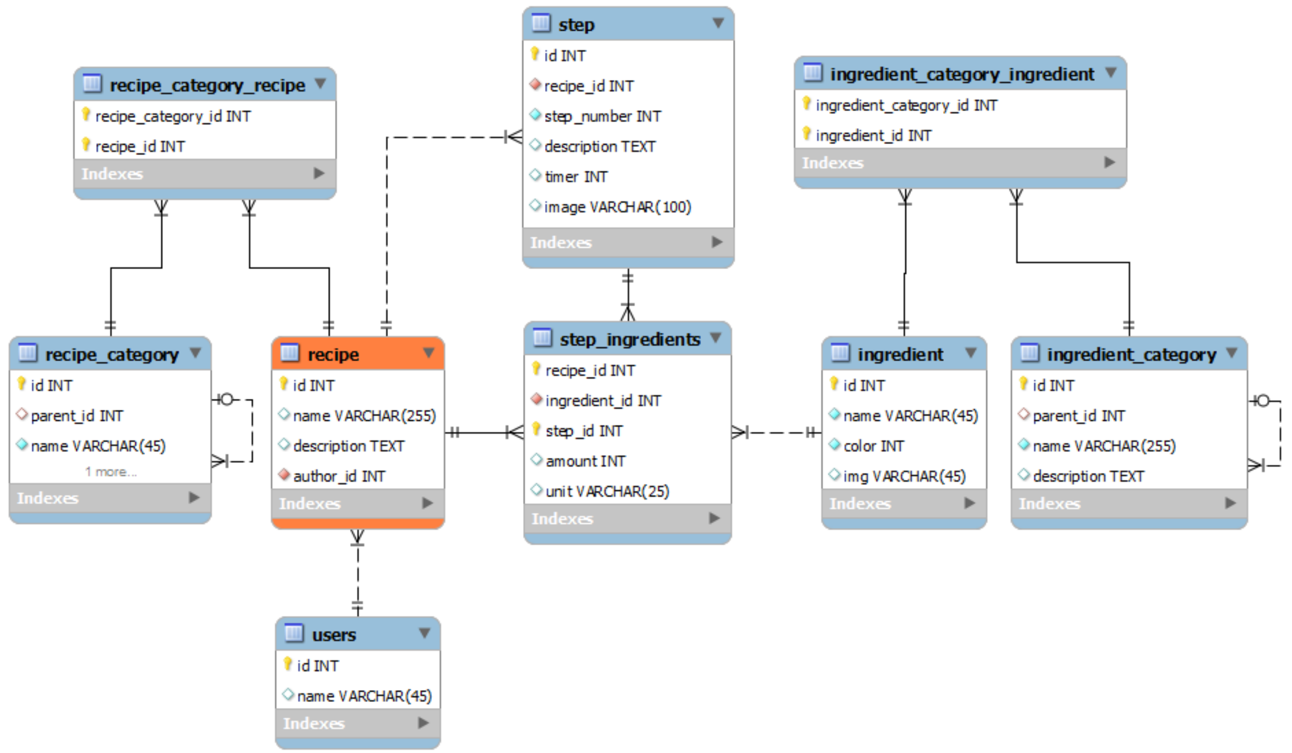
Next record

Move to last record

Number of records. An * indicates total not yet determined.

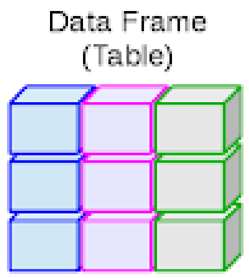
Click to find and replace records, select records by attributes, add fields, change the highlight color, add the table to the layout, export the table, and open related tables.

Image from ESRI

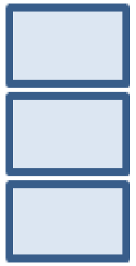


Data Frame

- Es una lista rectangular- todos sus elementos tienen la misma longuitud
- Se crean al usar read.csv() y read.table() - importar datos
- Se crea un nuevo data frame con data.frame()

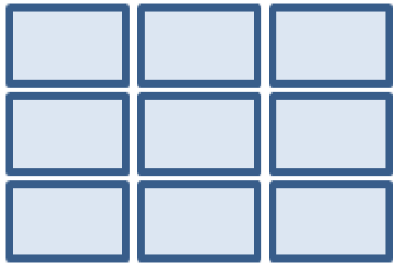


Vector



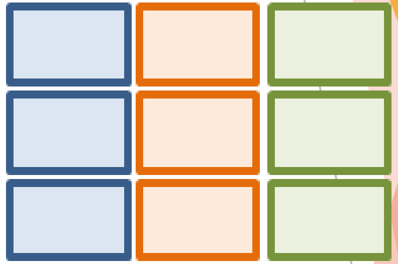
- 1 column or row of data
- 1 type (numeric or text)

Matrix



- multiple columns and/or rows of data
- 1 type (numeric or text)

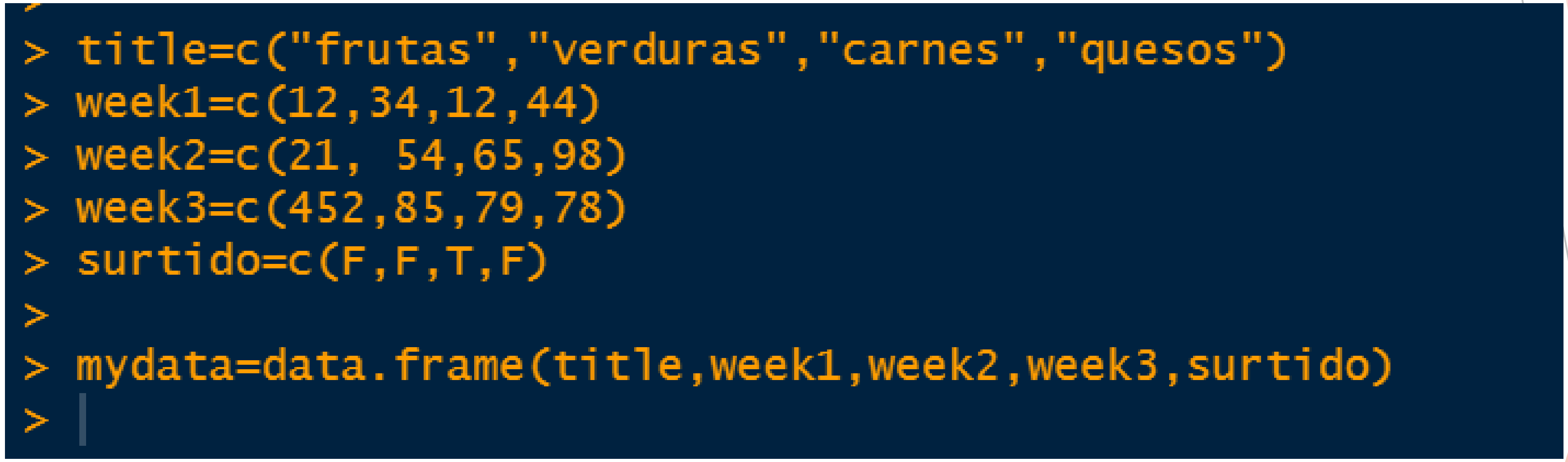
Data Frame



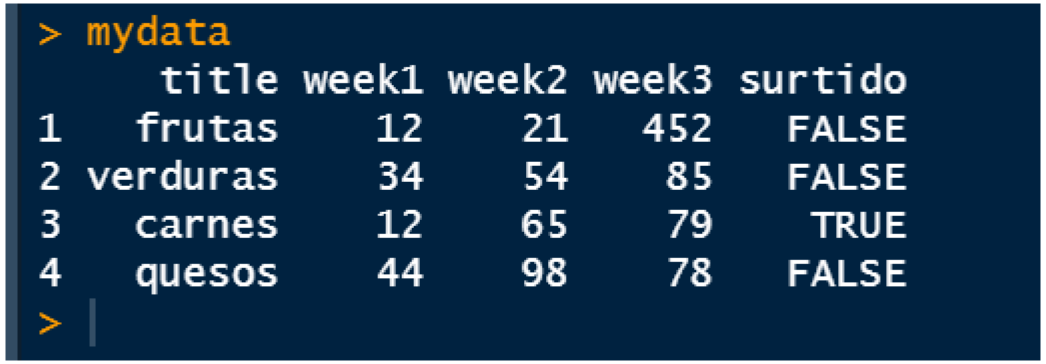
- multiple columns and/or rows of data
- multiple types

data.frame

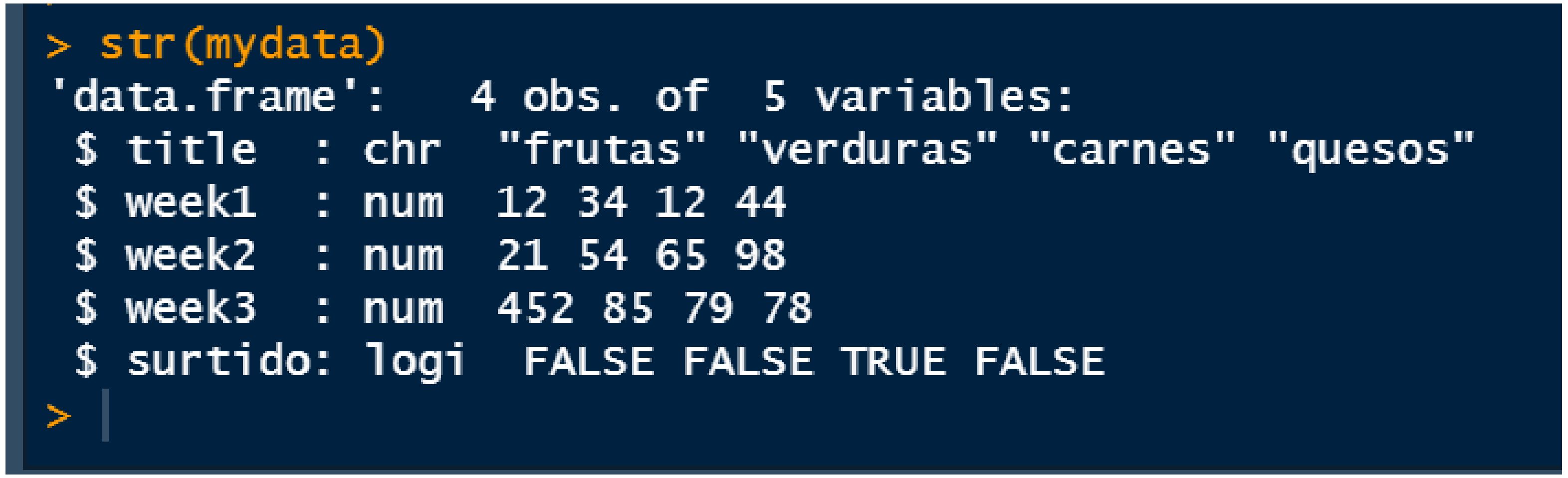
```
dat <- data.frame(id = letters[1:10], x = 1:10, y = 11:20)
```



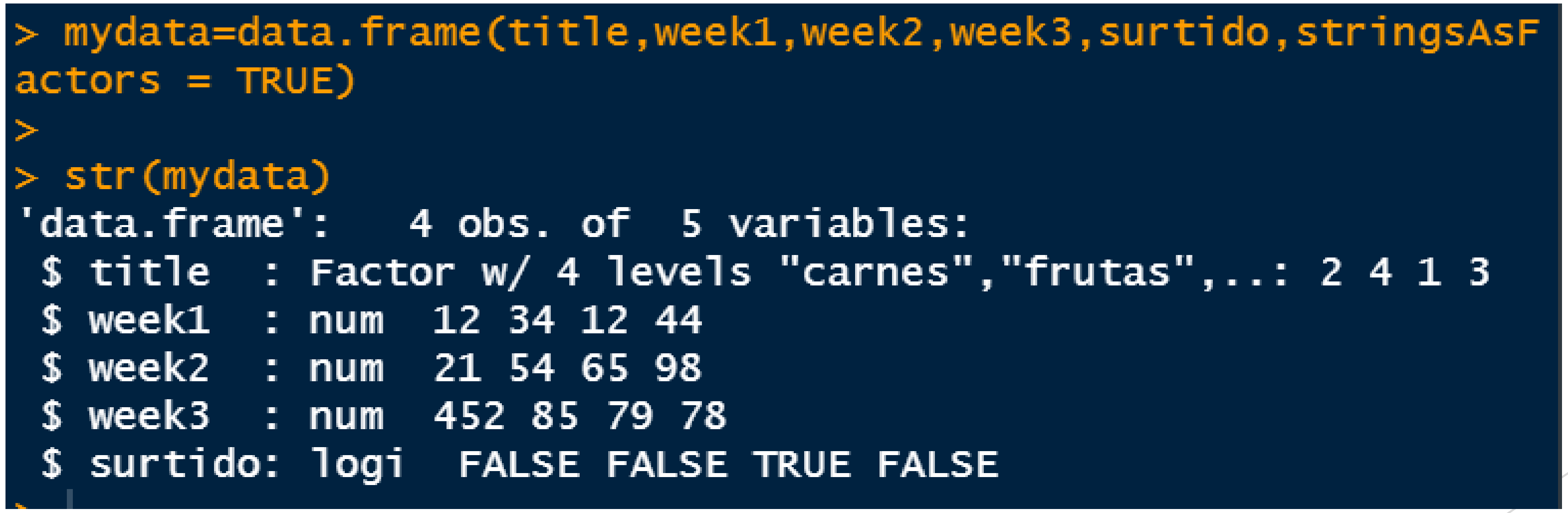
```
> title=c("frutas","verduras","carnes","quesos")
> week1=c(12,34,12,44)
> week2=c(21, 54,65,98)
> week3=c(452,85,79,78)
> surtido=c(F,F,T,F)
>
> mydata=data.frame(title,week1,week2,week3,surtido)
> 
```



```
> mydata
     title week1 week2 week3 surtido
1   frutas    12    21   452   FALSE
2 verduras    34    54    85   FALSE
3   carnes    12    65    79    TRUE
4   quesos    44    98    78   FALSE
> 
```



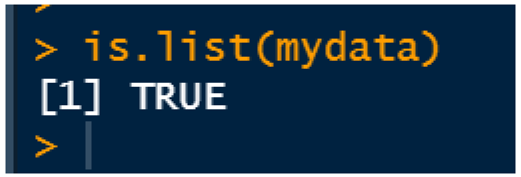
```
> str(mydata)
'data.frame':	4 obs. of  5 variables:
 $ title  : chr  "frutas" "verduras" "carnes" "quesos"
 $ week1  : num  12 34 12 44
 $ week2  : num  21 54 65 98
 $ week3  : num  452 85 79 78
 $ surtido: logi  FALSE FALSE TRUE FALSE
> 
```



```
> mydata=data.frame(title,week1,week2,week3,surtido,stringsAsF
actors = TRUE)
>
> str(mydata)
'data.frame':	4 obs. of  5 variables:
 $ title  : Factor w/ 4 levels "carnes","frutas",..: 2 4 1 3
 $ week1  : num  12 34 12 44
 $ week2  : num  21 54 65 98
 $ week3  : num  452 85 79 78
 $ surtido: logi  FALSE FALSE TRUE FALSE
```

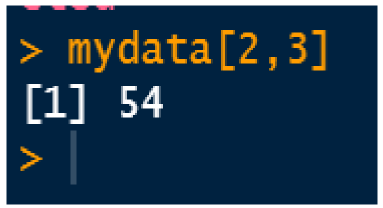
is.list

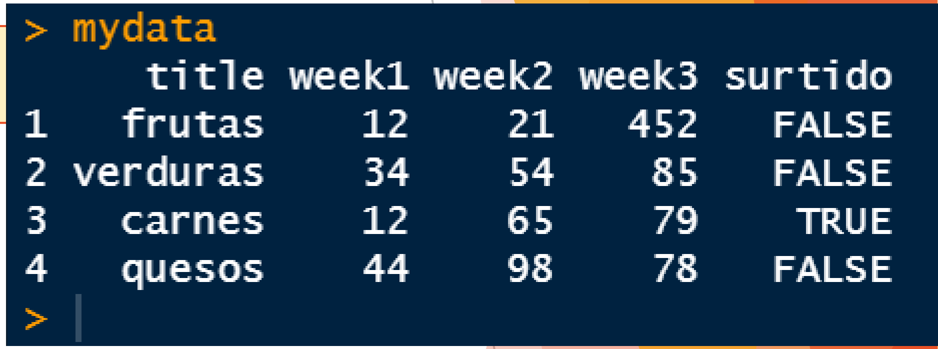
is.list(dat)


```
> is.list(mydata)
[1] TRUE
> 
```

dat[1, 3]

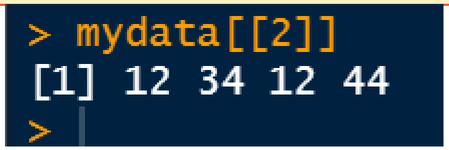
Accesar a un elemento

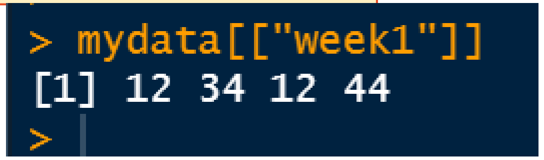

```
> mydata[2,3]
[1] 54
> 
```


```
> mydata
     title week1 week2 week3 surtido
1   frutas    12    21   452   FALSE
2 verduras    34    54    85   FALSE
3   carnes    12    65    79    TRUE
4   quesos    44    98    78   FALSE
> 
```

dat[["y"]]

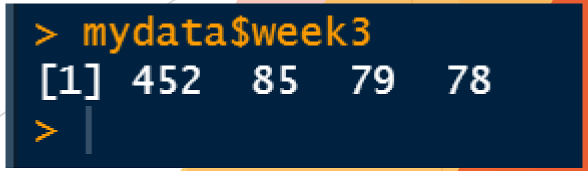
Accesar a una columna


```
> mydata[[2]]
[1] 12 34 12 44
> 
```


```
> mydata[["week1"]]
[1] 12 34 12 44
> 
```

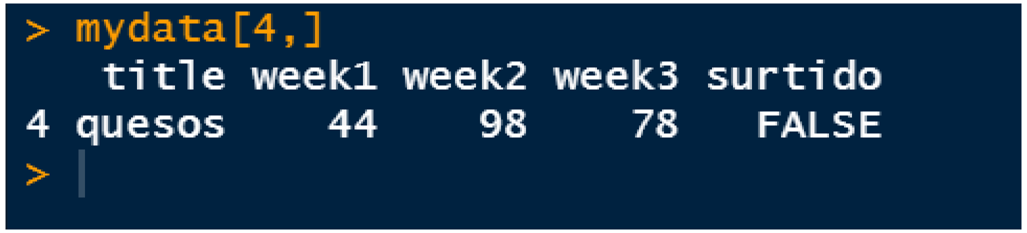
dat$y

Accesar a una columna

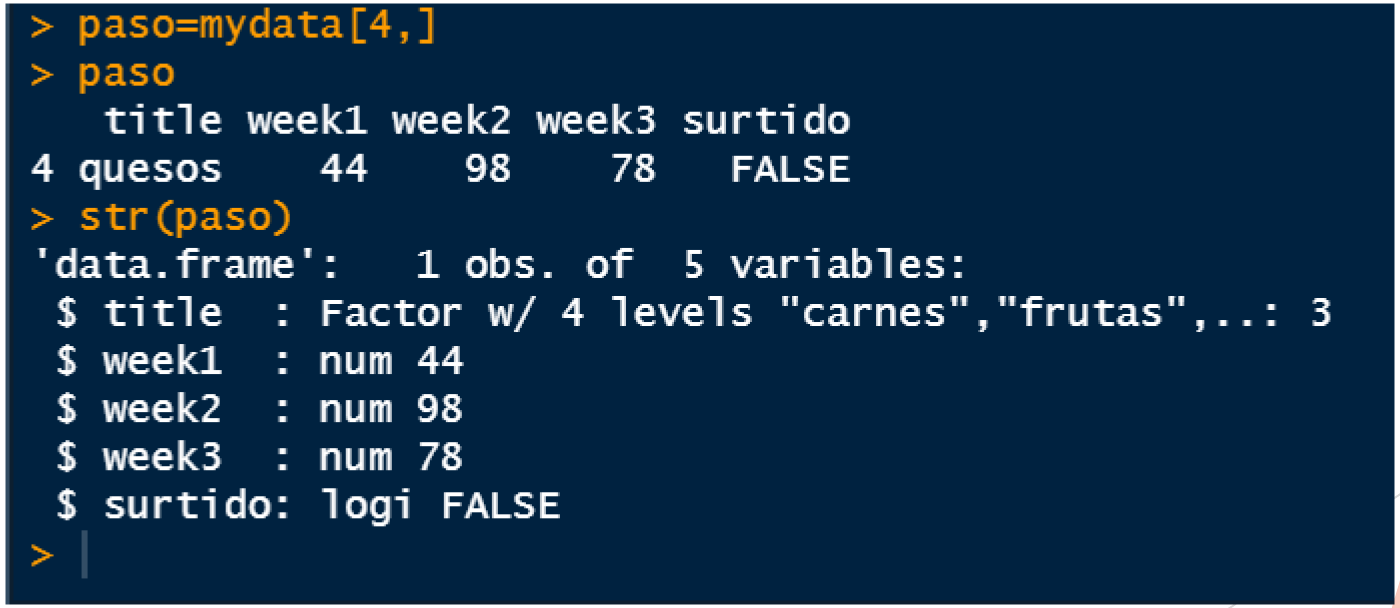

```
> mydata$week3
[1] 452  85  79  78
> 
```

dat[4,]

Accesar a un renglon



```
> mydata[4,]
   title week1 week2 week3 surtido
4 quesos    44    98    78   FALSE
> 
```



```
> paso=mydata[4,]
> paso
   title week1 week2 week3 surtido
4 quesos    44    98    78   FALSE
> str(paso)
'data.frame':	1 obs. of  5 variables:
 $ title  : Factor w/ 4 levels "carnes","frutas",..: 3
 $ week1  : num 44
 $ week2  : num 98
 $ week3  : num 78
 $ surtido: logi FALSE
> 
```

head()	Muestra los primeros 6 renglones
tail()	Muestra los últimos 6 renglones
dim()	Muestra la dimensión del data frame
nrow()	Numero de renglones
ncol()	Numero de columnas
str()	Estructura del data frame
names() / colnames	Nombres de cada columna