Manipulación de datos

..vamos a necesitar habilidades de programación

Revisión

- Escribir y guardar scripts en R & Rstudio
- Cómo cargar datos en R # read.csv ()
- Tipos y estructuras de datos comunes en R
- Tipos de datos: numeric, logical, factors, etc.
- Estructuras de datos: vectors, lists, matrices, data frames, etc.
- Cómo coaccionar datos y tipos de objetos # factor (), data.frame (), list ()

- Organizar los datos
  - Indexar
  - Pruebas lógicas
  - Otros trucos

Por qué??

¡Los conjuntos de datos vienen en muchos formatos!

¡Las funciones requieren un formato específico!

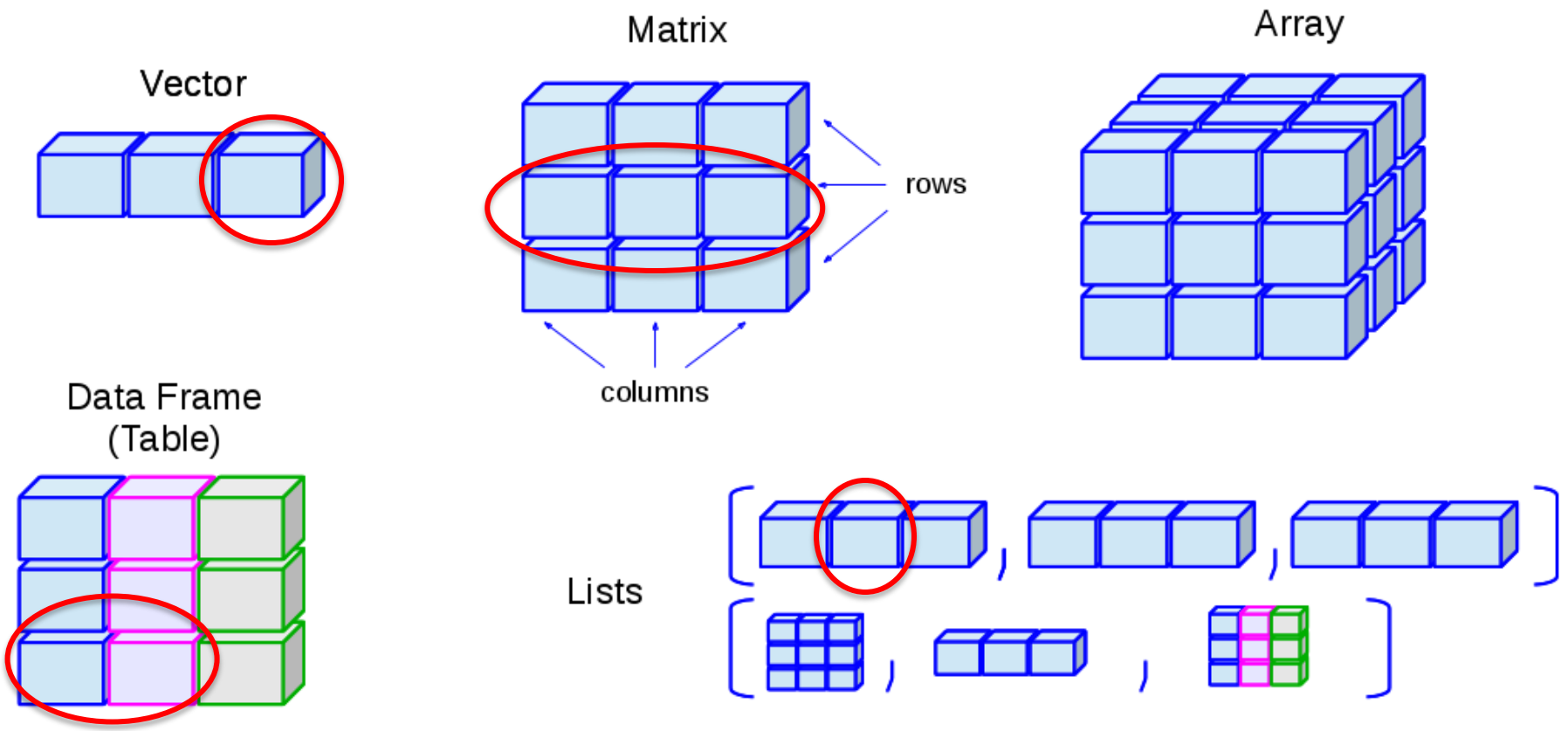
Objetivo:

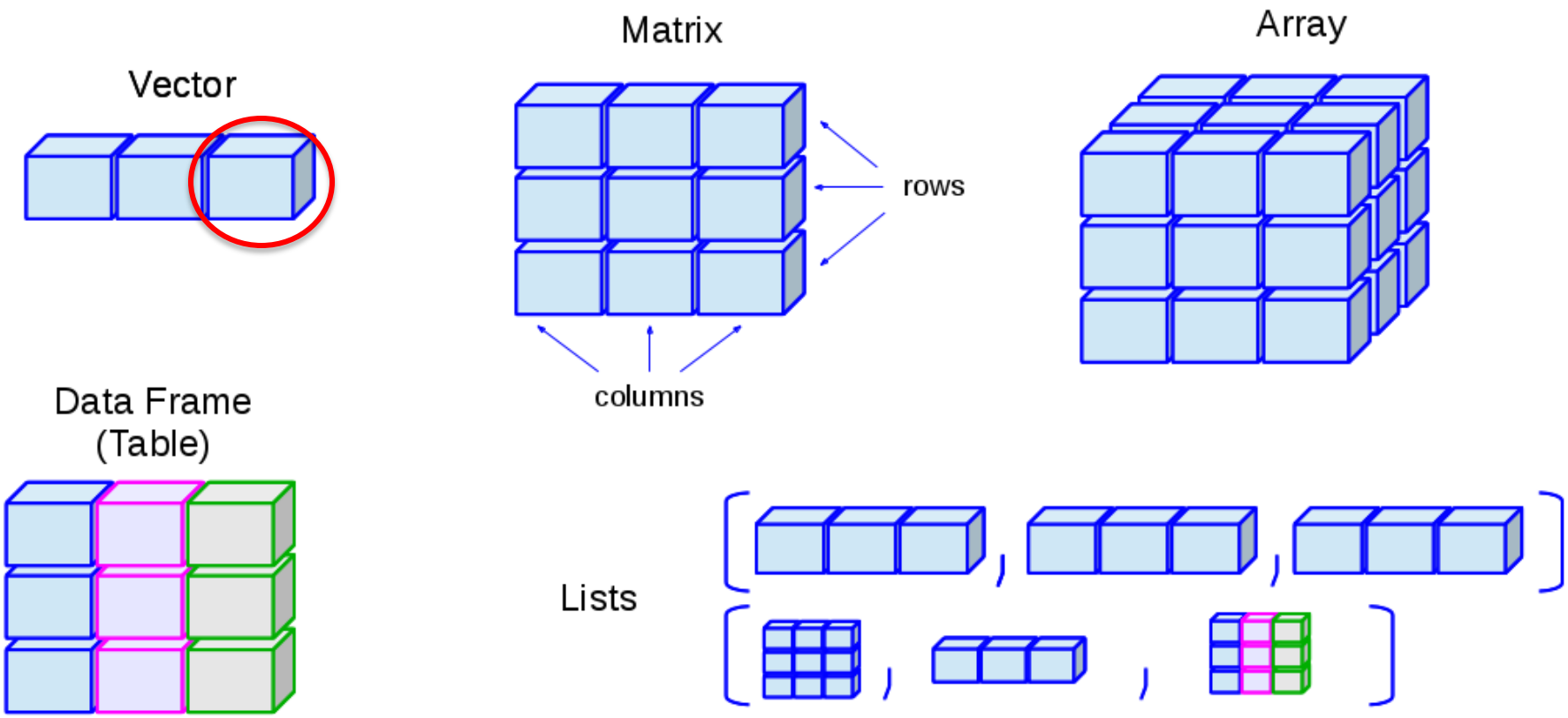
¡Hacer que los datos sean útiles para su análisis!

- Manipular y trabajar con los datos en R!
- Cosas que quizás quieras hacer: Limpia tus datos eliminando ciertas filas o columnas.
- Seleccionar subconjuntos de datos que cumplan con algunos criterios.
- Crear nuevas columnas basadas en los datos.
- Toma tiempo (50-80%?)

- Organizar los datos
  - **Indexar**
  - Pruebas lógicas
  - Otros trucos

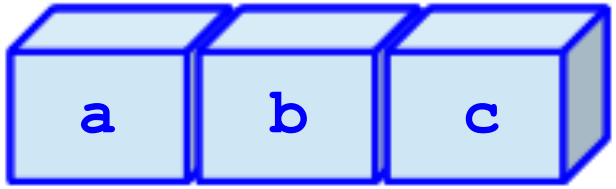
Indexación: subconjunto de un conjunto particular de valores




Vector
Matrix
rows
columns
Array
Data Frame
(Table)
Lists

1. Haz un vector

```
test <- c("a","b","c")
```

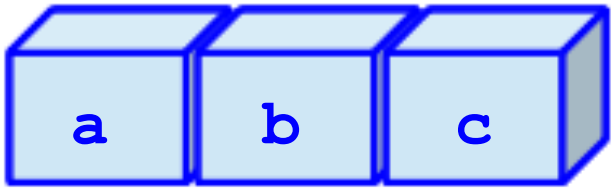


2. Para extraer el elemento x de vector, usar [x]

```
test[2]
[1] "b"
```

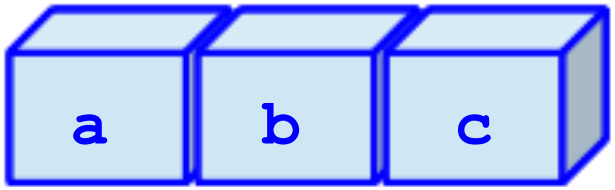
Como extraer c??

Prueben estos:

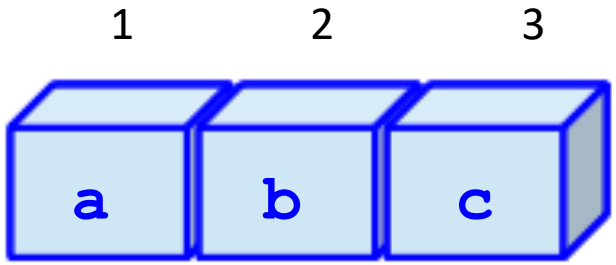



```
test[4]

test[-2]

test[1:2]

test[1]<- “d”
```

```
test[4]
NA

test[-2]
“a” “c”

test[1:2]
“a” ”b”

test[1]<- “d”
test
[1] "d" "b" "c"
```



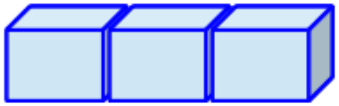
¡Puede usar la indexación para reorganizar el orden de sus datos!



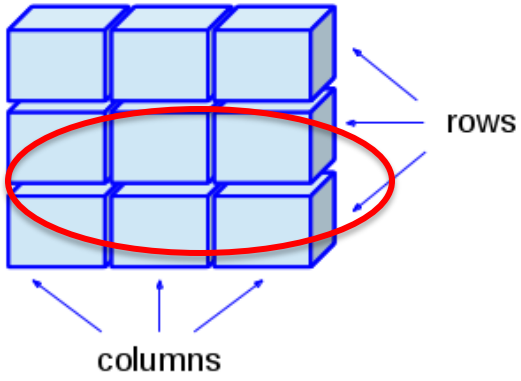

```
test[c(1,3,2)]
```

```
[1] a c b
```

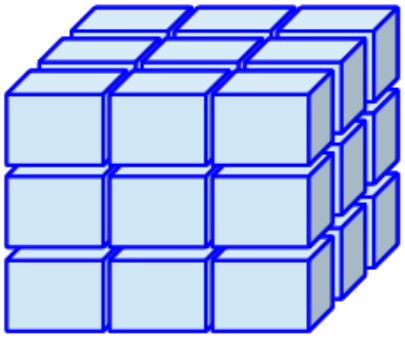
Vector



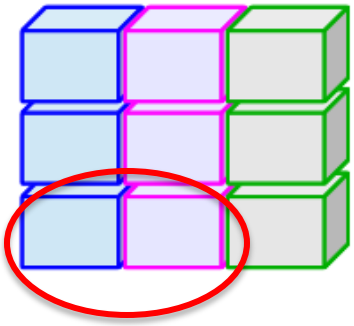
Matrix



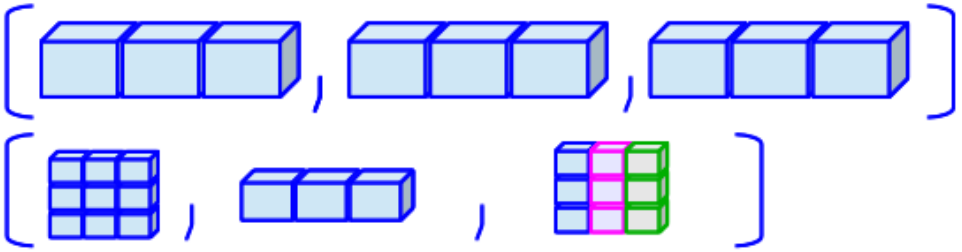
Array



Data Frame
(Table)



Lists



Matrix or dataframe

- Para extraer un elemento especifique el índice de fila, el índice de columna o ambos

```
object[row, column]
```

```
mtcars
dim(mtcars)
[1]32 11
```

```
mtcars[1,2]
```
Extract row 1 column 2

```
mtcars[1, ]
```
Extract first row, all columns (the whole first row)

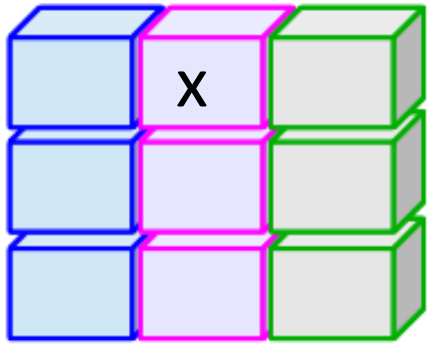
```
mtcars[ ,1:2]
```
Extract all rows, columns 1 through 2

Matrix or dataframe

- Para extraer un elemento especifique el índice de fila, el índice de columna o ambos

```
object[row, column]
```

```
mtcars
dim(mtcars)
[1]32 11
```




```
mtcars[1,2]
```
Extract row 1 column 2

```
mtcars[1, ]
```
Extract first row, all columns (the whole first row)

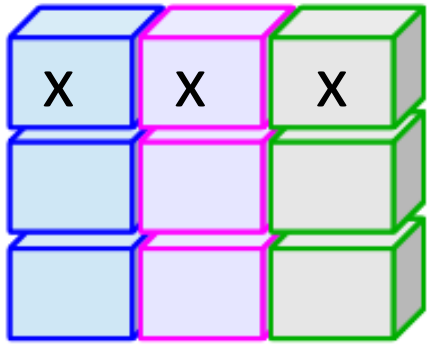
```
mtcars[ ,1:2]
```
Extract all rows, columns 1 through 2

Matrix or dataframe

- Para extraer un elemento especifique el índice de fila, el índice de columna o ambos

```
object[row, column]
```

```
mtcars
dim(mtcars)
[1]32 11
```




```
mtcars[1,2]
```
Extract row 1 column 2

```
mtcars[1, ]
```
Extract first row, all columns (the whole first row)

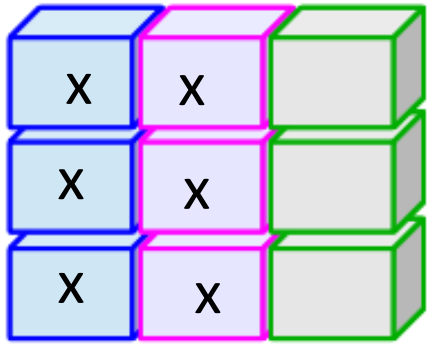
```
mtcars[ ,1:2]
```
Extract all rows, columns 1 through 2

Matrix or dataframe

- Para extraer un elemento especifique el índice de fila, el índice de columna o ambos

```
object[row, column]
```

```
mtcars
dim(mtcars)
[1]32 11
```

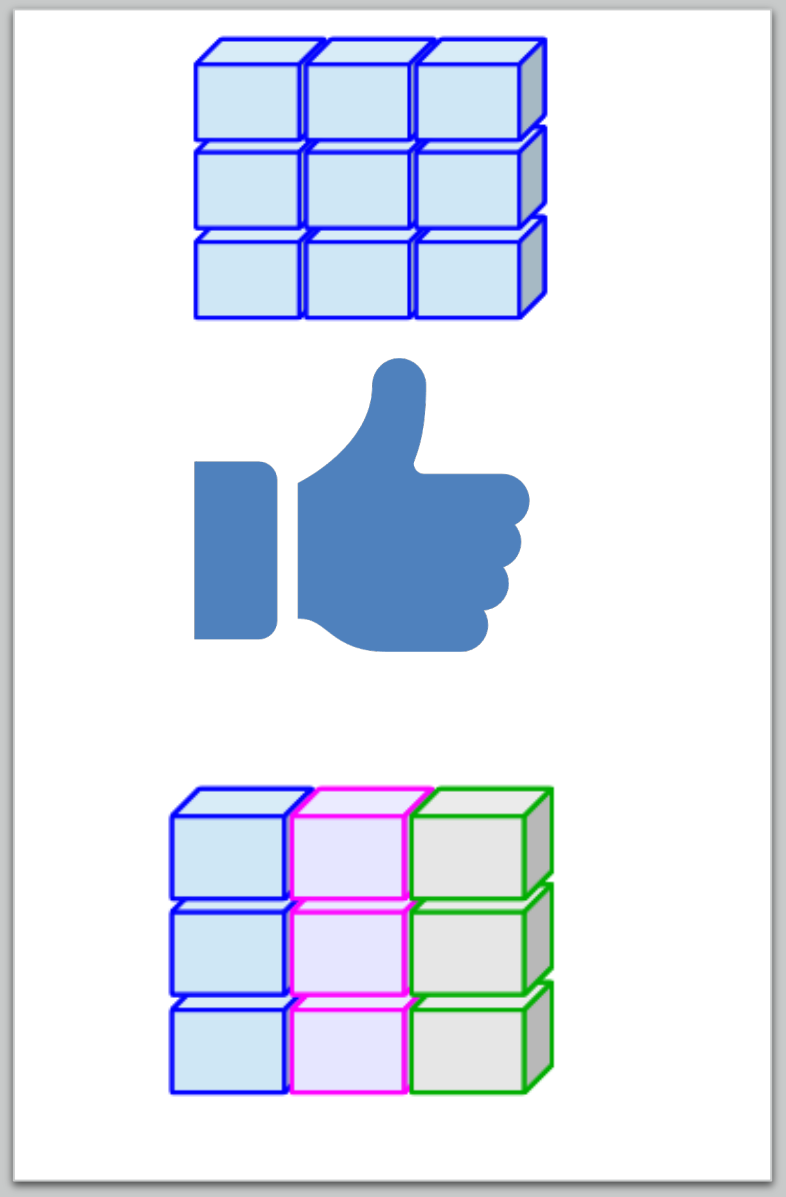



```
mtcars[1,2]
```
Extract row 1 column 2

```
mtcars[1, ]
```
Extract first row, all columns (the whole first row)

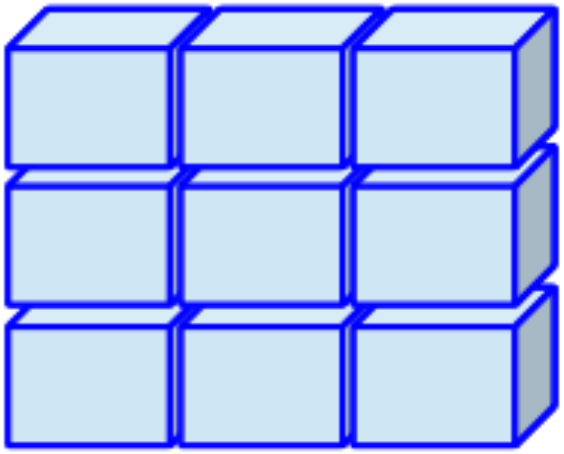
```
mtcars[ ,1:2]
```
Extract all rows, columns 1 through 2

```
> mtcars[ ,2:3]
                   cyl  disp
Mazda RX4            6 160.0
Mazda RX4 Wag        6 160.0
Datsun 710           4 108.0
Hornet 4 Drive       6 258.0 …

> mtcars[,c('cyl', 'disp')]
                   cyl  disp
Mazda RX4            6 160.0
Mazda RX4 Wag        6 160.0
Datsun 710           4 108.0
Hornet 4 Drive       6 258.0 …
```



Es conveniente almacenar datos como una colección de variables.

```
> x<- c(TRUE,FALSE,TRUE)
> y<- c("a","b","c")
> z<- c(4,5,7)
```



```
> my_matrix<-rbind(x, y, z)
  [,1]    [,2]     [,3]
x "TRUE"  "FALSE"  "TRUE"
y "a"     "b"      "c"
z "4"     "5"      "7"
```

```
> my_matrix<-cbind(x, y, z)
      x         y    z
[1,] "TRUE"   "a"  "4"
[2,] "FALSE"  "b"  "5"
[3,] "TRUE"   "c"  "7"
```

```
> my_dataframe<- data.frame(x, y, z)
```

Los nombres son muy utiles!

- Extraer nombres

```
> colnames(my_matrix) [1] "x" "y" "z"
```

- Modificar los nombres

```
> rownames(my_matrix) <- c(”monkey","bear”,”cow”)
> colnames(my_matrix) <- c(”scary","y”,”height”)
```

```
       scary y height
monkey  TRUE a      4
bear   FALSE b      5
cow     TRUE c      7
```

*Lo mismo para dataframe

```
> my_matrix[ ,"height"]
```

Extraer la columna "height"

```
> my_matrix["bear", ]
```

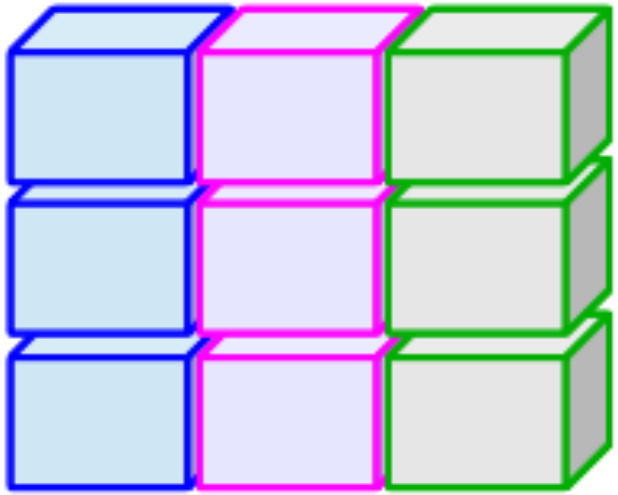
Extraer la fila "bear"

```
> my_matrix["bear", "height" ]
```

	scary	y	height
monkey	TRUE	a	4
bear	FALSE	b	5
cow	TRUE	c	7

*Lo mismo para dataframe

Data frames son mas especiales!

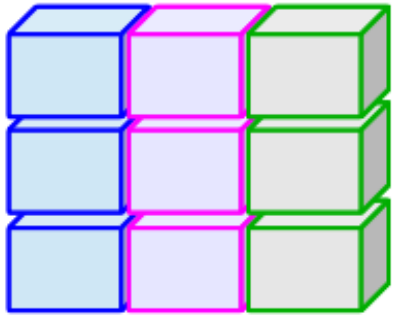


Extraer columnas de dataframe usando $

```
my_dataframe$
```

- scary
- y
- height

```
> my_dataframe$height
[1] 4 5 7
```



Para extraer la información (matrix o dataframe) especificar la fila y/o columna

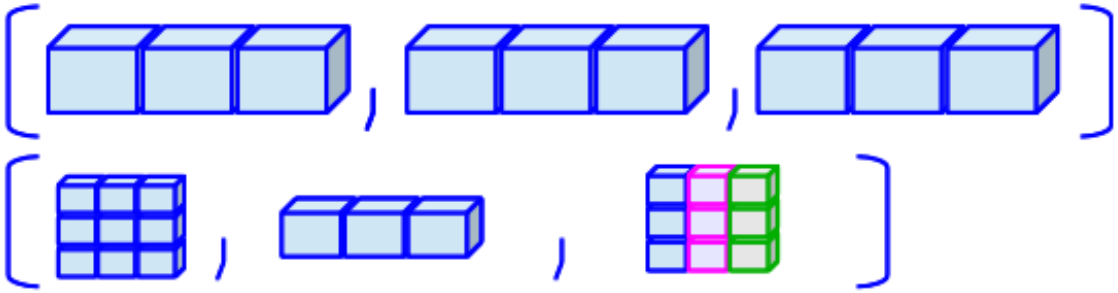
```
object[row, column]
```

Puede delimitar filas y columnas usando un vector de números o un vector con los nombres

```
object[1, 2] or object["bear", "height"]
```

Si es un dataframe pueden usar $

```
object$height or object$height[1]
```

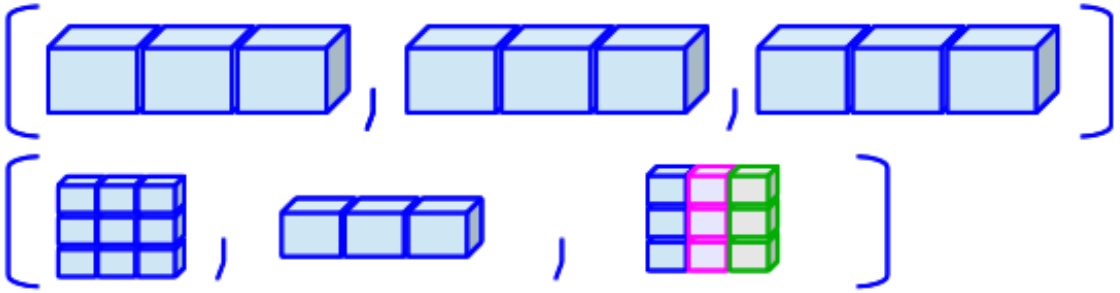


Para extraer datos de una lista usamos: [[ ]]

```
my_list
my_list[[1]]
```

Extrae el primer objeto de la lista

```
[[1]]
      x y z
1  TRUE a 4
2 FALSE b 5
3  TRUE c 7

[[2]]
[1] "universe"

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



Para extraer datos de una lista usamos: [[ ]]

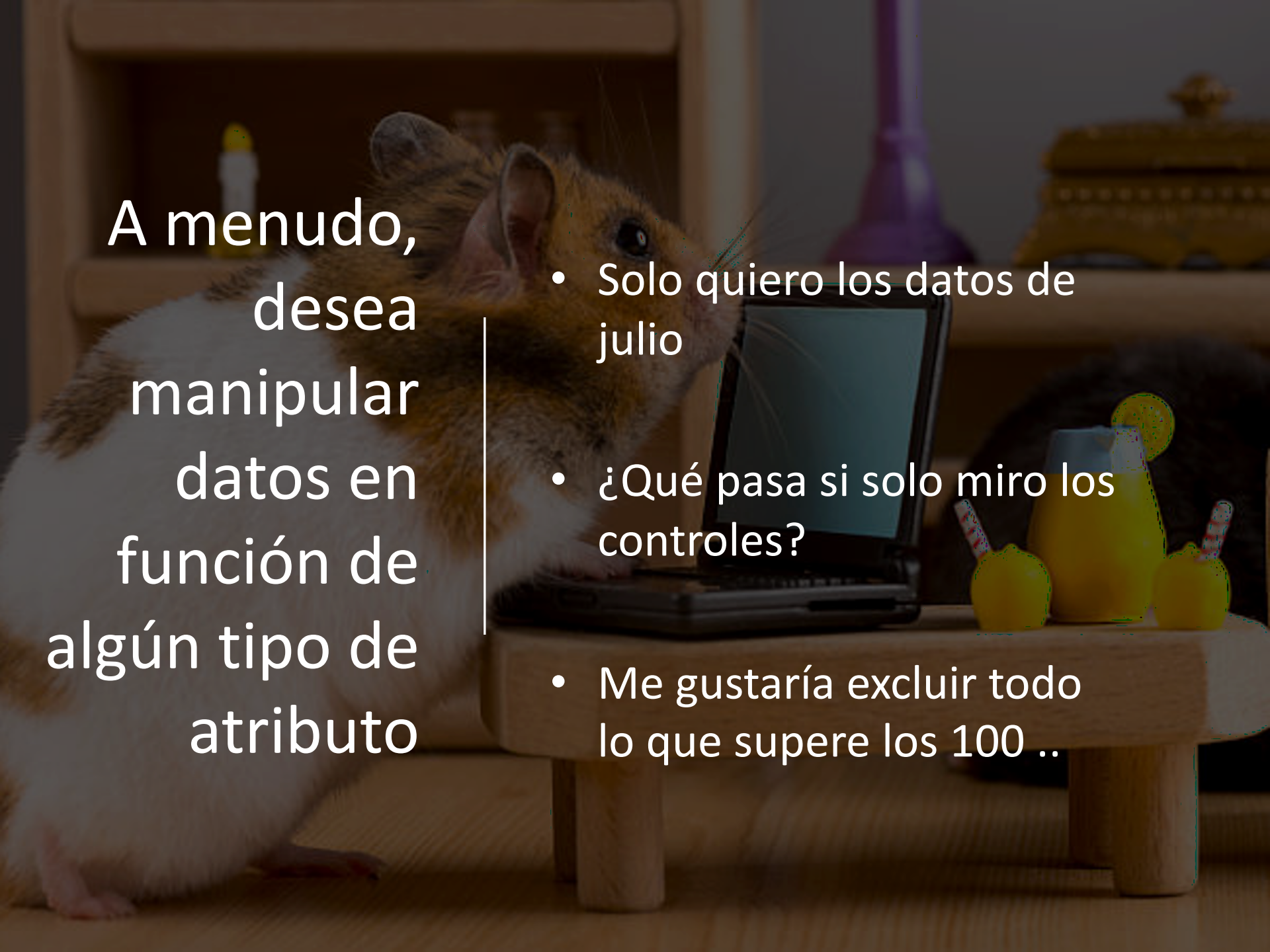
```
my_list
my_list[[1]]
```

Extrae el primer objeto de la lista

```
[[1]]
      x y z
1  TRUE a 4
2 FALSE b 5
3  TRUE c 7

[[2]]
[1] "universe"

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

```
my_list[[1]][2,1]
```

- Organizar los datos
  - Indexar
  - **Pruebas lógicas**
  - Otros trucos



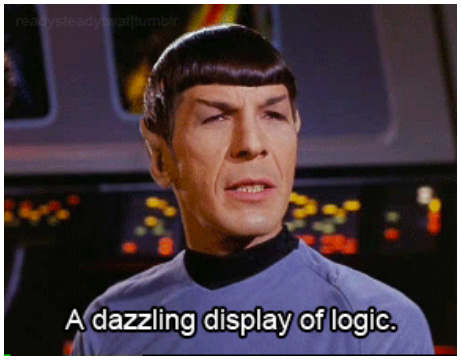
A menudo, desea manipular datos en función de algún tipo de atributo

- Solo quiero los datos de julio
- ¿Qué pasa si solo miro los controles?
- Me gustaría excluir todo lo que supere los 100 ..

Usa la lógica!

Podemos usar operadores de comparación para crear vectores lógicos.

```
a <-c(1,2,3,4,5)
[1] 1 2 3 4 5

b <- a>3

[1] FALSE FALSE FALSE TRUE TRUE
```



Ahora los puedes usar como índices!

```
a[b] #solo los valores >3
[1] 4 5
```

Comparaciones

<	menor que
>	mayor que
<=	menor o igual
>=	mayor o igual
==	igual ← = != == = == <-
!	no
!=	no es igual
&	y
|	o

Intenten esto!

```
> x <- 3

> x == 3


> x < 10


> x < -1


> x > 0 & x < 10
```

```
> x <- 3

> x == 3
[1] TRUE

> x < 10
[1] TRUE

> x < -1
[1] FALSE

> x > 0 & x < 10
[1] TRUE
```

Combinar comparaciones: y (&) o (|)

Así se pregunta si x está entre 0 y 10

Ahora intenten esto:

```
> x <- 1:5

> x == 3


> x < 10


> x > 2 & x <= 4


> x != 2
```

```
> x <- 1:5

> x == 3
[1] FALSE FALSE TRUE FALSE FALSE

> x < 10
[1] TRUE TRUE TRUE TRUE TRUE

> x > 2 & x <= 4
[1] FALSE FALSE TRUE TRUE FALSE

> x != 2
[1] TRUE FALSE TRUE TRUE TRUE
```

Pro tips: TRUE and FALSE

T

F

tip 1:

En R, F es equivalente a False y T es equivalente a TRUE.

tip 2:

La representación de TRUE y FALSE es 0 y 1

TRUE == T == 1

FALSE == F == 0

```
> T+T
[1] 2
```

```
> T+T
[1] 2
```

TRUE == T == 1
FALSE == F == 0

```
> rain <- c("Yes","Yes","Yes","Yes","Yes","Yes","No")
```

Cuantos días llovió?

```
> sum(rain=="Yes")
[1] 6
```

```
> any(rain=="Yes")
[1] TRUE
```

Otros trucos con operadores lógicos

```
> rain <- c("Yes","Yes","Yes","Yes","Yes","Yes","No")
```

Todos los elementos == “Yes”?

```
> all(rain=="Yes")
[1] FALSE
```

Cuales elementos == “Yes”?

```
> which(rain=="Yes")
[1] 1 2 3 4 5 6
```

Cuantos días llovió?

```
> sum(rain=="Yes")
[1] 6

> any(rain=="Yes")
[1] TRUE
```

Todos los elementos == “Yes”?

```
> all(rain=="Yes")
[1] FALSE
```

Cuales elementos == “Yes”?

```
> which(rain=="Yes")
[1] 1 2 3 4 5 6
```

- Organizar los datos
  - Indexar
  - Pruebas lógicas
  - **Otros trucos**

Función sort( )

- A menudo queremos reorganizar un conjunto de datos por una sola variable

```
sort()

cards <- sample(1:10)
[1]  6  5  3 10  7  9  1  8  4  2

sort(cards)
[1]  1  2  3  4  5  6  7  8  9 10

rev(sort(cards))
[1] 10  9  8  7  6  5  4  3  2  1
```

Manipulación de datos
Datos perdidos (NA)

```
humidity <- c(63.33, NA, 64.63, 68.38, NA,
79.1, 77.46)
```

NA = not available

Muchas funciones no funcionan con NA

```
mean(humidity)
[1] NA
```

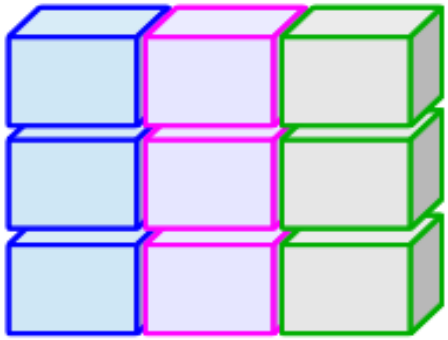
```
mean(humidity, na.rm=T)
[1] 70.58
```

Argumento para remover NAs antes de usar la función

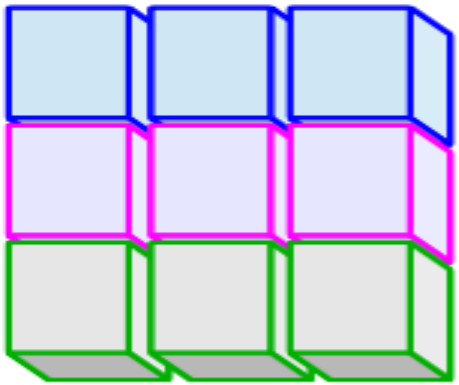
Manipulación de datos – t()

- Use `t()` para transponer (cambiar columnas y filas)

```
> my_dataframe

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



```
> t(my_dataframe)

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



Manipulación de datos – unique()

```
> plates <- c("WA","WA","OR","RI","WA","WA","CA”)
```

Valores únicos

```
> unique(plates)
[1] "WA" "OR" "RI" "CA"
```

Cuales elementos están duplicados

```
> duplicated(plates)
[1] FALSE TRUE FALSE FALSE TRUE TRUE FALSE
```

Manipulación de datos – length()

Esta función devuelve el número de elementos en un vector.

```
> length(islands)
[1] 48

> nislands <- length(islands)
> 1:nislands
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47 48
```

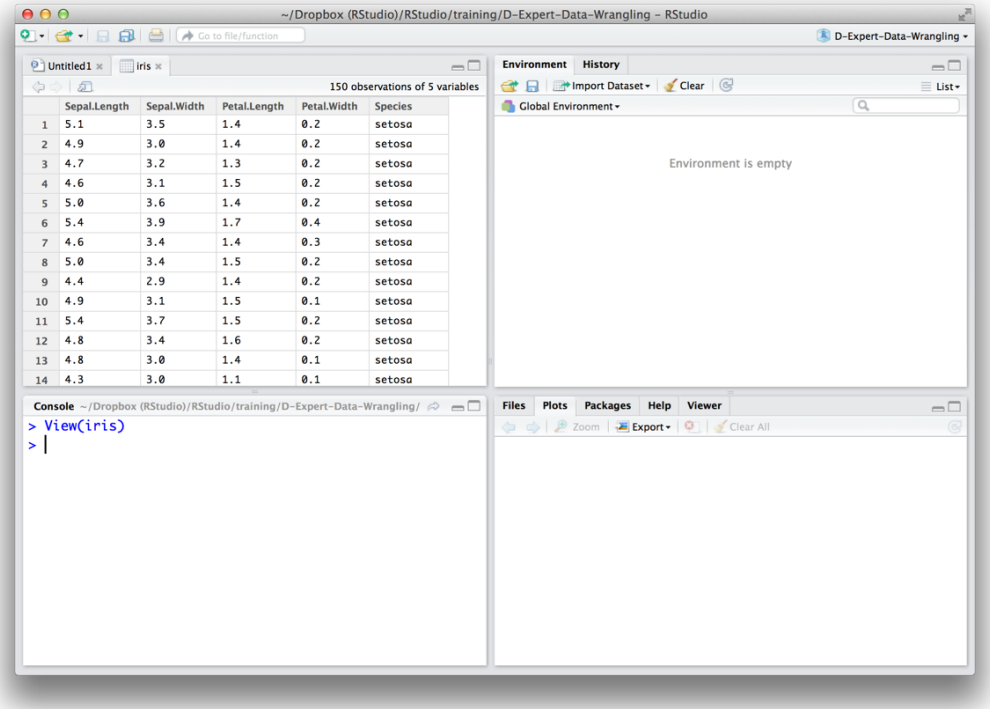
Manipulación de datos – subset()

```
> myDataFrame <-
    data.frame(a=c(11,13,12,15,17,20),
               b=c(8,NA, 6, 4,NA,15))

> subset(x=myDataFrame, subset=b>5)
   a  b
1 11  8
3 12  6
6 20 15

> subset(x=myDataFrame, subset=b>7, select=a)
   a
1 11
6 20
```

Manipulación de datos

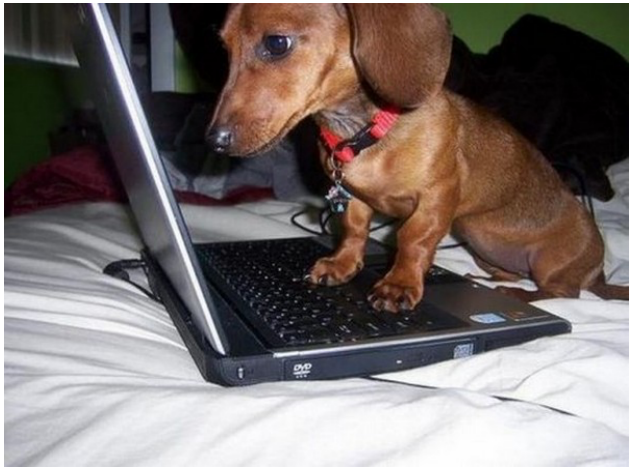
View( )

```
View(iris)
View(mtcars)
View(pressure)
```



Lo que aprendimos!

- Como indexar diferentes tipos de objetos
- Usar lógica para indexar
- Remover NA para los análisis.



- Cómo ordenar, transponer, eliminar duplicados, etc.