Working with dataframes

Ottavia M. Epifania, Ph.D

Lezione di Dottorato @Università Cattolica del Sacro Cuore (MI)

8-9 Giugno 2023

Table of contents

Table of Contents

Superato lo scoglio dell'importazione dei dati è tutta in discesa (kind of)

Diversi tipi di formati:

- `.csv` comma separatered value $\rightarrow$ sono tra i più usati, "universali" (nonché i miei preferiti)
- `.tab` o `.dat` plain text, li potete creare anche con il blocco note del computer, molto comodi
- `.xls` o `.xlsx` molto comuni, servono pacchetti esterni
- `.sav` servono pacchetti esterni

`.csv`

`comma separatered value` → i separatori di colonna sono le virgole “,”

Nonostante siano `comma separatered value` nei computer in italiano sono “;”, cosa che ovviamente genera non poca confusione

Il comando di base per leggere i `.csv`:

```
read.csv(file, header = TRUE, sep = ",", quote = "\",
         dec = "."", ...)
```

`file`: Il nome del file (se serve anche la sua directory)

`header = TRUE`: La prima riga contiene i nomi delle variabilii

`sep = ","`: I separatori delle colonne sono le virgole

`dec = "."`: Il separatore dei decimali

`.csv` in Italia

Due opzioni:

1. usare la funzione `read.csv = sep()` settando `sep = ";"` e `dec=","`
2. usare la funzione `read.csv2()` $\rightarrow$ cambiano i default per cui `sep =";"` e dec = ","

`read.csv()` in pratica

Come si chiama il file?

```
dir("data") # elenca tutti gli oggetti che sono  all'interno
```

```
[1] "babies.tab" "benessere.csv" "CioccoRazzaBuilt.dat"
[4] "database_benessere.xls" "datiBenessere.xlsx"
```

Voglio importare il data set `benessere.csv` e assegnarlo all'oggetto `data`:

```
data = read.csv("data/benessere.csv",
                header = TRUE,
                sep =",", dec = ".")
```

Ha funzionato?

Si!

```
head(data)
```

```
  benessere stipendio genere
1         5 1461.0983      m
2         7 1132.3637      f
3         7 1675.9004      m
4         2  328.9587      f
5         6 1370.0146      m
6         5  954.3540      f
```

Ha funzionato?

No

```
head(data.2)
```

```
  benessere.stipendio.genere
1          5,1461.09828023079,m
2          7,1132.36368361099,f
3          7,1675.90040479853,m
4          2,328.958701913838,f
5          6,1370.01460952768,m
6          5,954.354030915469,f
```

.tab o .dat

Il comando di base per leggere i `.tab` o `.dat`:

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", ...)
```

`file`: Il nome del file (se serve anche la sua directory)

`header = FALSE`: La prima riga contiene i nomi delle variabili (letto di default)

`sep = ""`: I separatori delle colonne sono inferiti dal file

`dec = "."`: Il separatore dei decimali

read.table() in pratica I

```
tab_data = read.table("data/babies.tab")
head(tab_data)
```

```
     id genere       peso  altezza
1 baby1      f  7.424646 62.07722
2 baby2      m  7.442727 58.18877
3 baby3      f  9.512598 84.52737
4 baby4      f 11.306349 85.13573
5 baby5      m  9.345165 75.23783
6 baby6      m  5.411290 46.80163
```

read.table() in pratica II

```
dat_data = read.table("data/CioccoRazzaBuilt.dat",
                      header = TRUE, sep = "\t")
head(dat_data)
```

```
date time build subject blocknum blockcode trialnum
1 121318 09:55 3.0.6.0 1 1 consenso 1
2 121318 09:55 3.0.6.0 1 2 fame 1
3 121318 09:55 3.0.6.0 1 6 BuiltLatteFondente2nd 1
4 121318 09:55 3.0.6.0 1 6 BuiltLatteFondente2nd 2
5 121318 09:55 3.0.6.0 1 6 BuiltLatteFondente2nd 4
6 121318 09:55 3.0.6.0 1 6 BuiltLatteFondente2nd 6
trialcode
1 consenso
2 fame
3 reminder
4 Builtlatteright
5 Builtfondenteleft
6 Builtfondenteleft
response correct
```

File excel

Servono aiuti esterni (un pacchetto apposito):

`install.packages("readxl")` va digitato e fatto correre una volta nella console

```
library("readxl") # rende disponibile il pacchetto nell'ambie
```

La funzione:

```
read_excel(path, sheet = NULL, range = NULL, col_names = TRUE
           col_types = NULL, ...)
```

`path`: Il nome del file (se serve anche la sua directory)

`sheet = NULL` dà la possibilità di speicificare il foglio specifico del file excel

`range = NULL`: Il range specifico di celle da leggere (e.g., `B0:B13`)

`col_names = TRUE`: La prima riga contiene i nomi delle variabili

read_excel() in pratica

```
benessere = read_excel("data/datiBenessere.xlsx")
head(benessere)
```

```
# A tibble: 6 x 19
ID età genere frat item1 item2 item3 item4 item5 au1 au2
au3 au4
<dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
<dbl> <dbl> <dbl>
1 1 16 1 0 1 2 4 3 4 5 4 5 2
2 2 21 1 1 2 2 3 4 3 5 4 5 2
3 3 28 2 4 2 3 5 1 2 4 4 4 4
4 4 15 2 2 3 4 2 2 2 4 5 5 3
5 5 23 1 3 4 5 1 5 2 3 2 3 2
6 6 31 1 2 2 3 2 1 2 5 1 5 1
# i 6 more variables: au5 <dbl>, au6 <dbl>, au7 <dbl>, au8
<dbl>, au9 <dbl>,
# au10 <dbl>
```

File .sav

Aiuti esterni:

`install.packages("foreign")` oppure
`install.packages(foreign)`

Table of Contents

Sorting

`order()`:

```
babies[order(babies$peso), ] # sort by increasing peso
```

```
       id genere      peso  altezza
5   baby5      m  5.547482 50.95574
10 baby10      f  6.899776 71.39348
2   baby2      f  8.128073 72.30143
8   baby8      f  8.638523 83.48096
9   baby9      m  8.767017 65.63333
1   baby1      m  9.858097 68.83238
6   baby6      m 10.005233 70.75147
3   baby3      f 11.347780 74.75981
4   baby4      f 15.246266 87.40951
7   baby7      m 15.253421 89.57014
```

```
babies[order(babies$peso,    # sort by decreasing peso
             decreasing = T), ]
```

Multiple arguments in `order`:

Aggregating

Aggregate a response variable according to grouping variable(s) (e.g., averaging per experimental conditions):

```
# Single response variable, single grouping variable
aggregate(y ~ x, data = data, FUN, ...)

# Multiple response variables, multiple grouping variables
aggregate(cbind(y1, y2) ~ x1 + x2, data = data, FUN, ...)
```

Aggregating: Example

```
ToothGrowth # Vitamin C and tooth growth (Guinea Pigs)
```

```
    len supp dose
1   4.2   VC  0.5
2  11.5   VC  0.5
3   7.3   VC  0.5
4   5.8   VC  0.5
5   6.4   VC  0.5
6  10.0   VC  0.5
7  11.2   VC  0.5
8  11.2   VC  0.5
9   5.2   VC  0.5
10  7.0   VC  0.5
11 16.5   VC  1.0
12 16.5   VC  1.0
13 15.2   VC  1.0
14 17.3   VC  1.0
15 22.5   VC  1.0
16 17.3   VC  1.0
```

Reshaping: Long to wide

Data can be organized in wide format (i.e., one line for each statistical unit) or in long format (i.e., one line for each observation).

```
Indometh # Long format
```

```
   Subject time conc
1        1 0.25 1.50
2        1 0.50 0.94
3        1 0.75 0.78
4        1 1.00 0.48
5        1 1.25 0.37
6        1 2.00 0.19
7        1 3.00 0.12
8        1 4.00 0.11
9        1 5.00 0.08
10       1 6.00 0.07
11       1 8.00 0.05
```

Long to wide

```
# From long to wide
df.w <- reshape(Indometh, v.names = "conc", timevar = "time",
    idvar = "Subject", direction = "wide")
```

```
   Subject conc.0.25 conc.0.5 conc.0.75 conc.1 conc.1.25 conc.2 con
1        1      1.50     0.94      0.78   0.48       0.37   0.19   0
12       2      2.03     1.63      0.71   0.70       0.64   0.36   0
23       3      2.72     1.49      1.16   0.80       0.80   0.39   0
34       4      1.85     1.39      1.02   0.89       0.59   0.40   0
45       5      2.05     1.04      0.81   0.39       0.30   0.23   0
56       6      2.31     1.44      1.03   0.84       0.64   0.42   0
   conc.5 conc.6 conc.8
1    0.08   0.07   0.05
12   0.25   0.12   0.08
23   0.11   0.08   0.08
34   0.10   0.07   0.07
45   0.08   0.10   0.06
56   0.13   0.10   0.09
```

Reshaping: Wide to long

```
# From wide to long
df.l <- reshape(df.w, varying = list(2:12), v.names = "conc",
    idvar = "Subject", direction = "long", times = c(0.25, 0.5,
        0.75, 1, 1.25, 2, 3, 4, 5, 6, 8))
```

```
        Subject time conc
1.0.25        1 0.25 1.50
2.0.25        2 0.25 2.03
3.0.25        3 0.25 2.72
4.0.25        4 0.25 1.85
5.0.25        5 0.25 2.05
6.0.25        6 0.25 2.31
1.0.5         1 0.50 0.94
2.0.5         2 0.50 1.63
3.0.5         3 0.50 1.49
4.0.5         4 0.50 1.39
5.0.5         5 0.50 1.04
6.0.5         6 0.50 1.44
1.0.75        1 0.75 0.78
```

Table of Contents

Vengono creati **c**oncatenando Impo variabili insieme

Si usa la funzione `c()`

Tutte le variaili all'interno della funzione `c()` vanno separate da una virgola

Diversi tipi di variabili $\rightarrow$ diversi tipi di vettori:

- `int`: vettori numerici (numeri interi)
- `num`: vettori numerici (numeri continui)
- `logi`: vettori logici
- `chr`: vettori character
- `factor`: vettori factor con diversi livelli

int & num

`int`: numeri interi: -3, -2, -1, 0, 1, 2, 3

```
mesi = c(5, 6, 8, 10, 12, 16)
```

```
[1]  5  6  8 10 12 16
```

`num`: tutti i valori numerici tra $-\infty$ e $+\infty$: 0.2686296, -0.3527401, -0.619082, 1.1779617, 0.3604757, 0.0473789

```
peso = seq(3, 11, by = 1.5)
```

```
[1]  3.0  4.5  6.0  7.5  9.0 10.5
```

logi

Valori logici possono essere veri `TRUE` (`T`) o falsi `FALSE` (`F`):

```
v_logi = c(TRUE, TRUE, FALSE, FALSE, TRUE)
```

```
[1]  TRUE  TRUE FALSE FALSE  TRUE
```

Si usano per testare delle condizioni:

```
mesi > 12
```

```
[1] FALSE FALSE FALSE FALSE FALSE  TRUE
```

chr & factor

chr: characters: a, b, c, D, E, F

```
v_chr = c(letters[1:3], LETTERS[4:6])
```

```
[1] "a" "b" "c" "D" "E" "F"
```

factor: Usa numeri o carattarri per identificare i livelli della variabile:

```
ses = factor(c(rep(c("low", "medium", "high"), each = 2)))
```

```
[1] low    low    medium medium high   high
Levels: high low medium
```

Si può cambaire l'ordine dei livelli:

```
ses1 = factor(ses, levels = c("medium", "high", "low"))
```

```
[1] low    low    medium medium high   high
Levels: medium high low
```

Creare i vettori

Concatenare le variabili con `c()`: `vec = c(1, 2, 3, 4, 5)`

Utilizzando le sequenze:

```
-5:5 # vector of 11 numbers from -5 to 5
```

```
 [1] -5 -4 -3 -2 -1  0  1  2  3  4  5
```

```
seq(-2.5, 2.5, by = 0.5) # sequence in steps of 0.5
```

```
 [1] -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5
```

Ripetendo gli elementi:

```
rep(1:3, 4)
```

```
 [1] 1 2 3 1 2 3 1 2 3 1 2 3
```

Creare i vettori II

```
rep(c("condA", "condB"), each = 3)
```

```
[1] "condA" "condA" "condA" "condB" "condB" "condB"
```

```
rep(c("on", "off"), c(3, 2))
```

```
[1] "on"  "on"  "on"  "off" "off"
```

```
paste0("item", 1:4)
```

```
[1] "item1" "item2" "item3" "item4"
```

Non mischiate i vettori! a meno che non lo vogliate davvero

`int` + `num` → `num`

`int/num` + `logi` → `int/num`

`int/num` + `factor` → `int/num`

`int/num` + `chr` → `chr`

`chr` + `logi` → `chr`

Vettori e operazioni

I vettori possono essere sommati/divisi/moltiplicati tra di loro o anche per un numero singolo

```
a = c(1:8) # vettore di lunghezza 8
a
```

```
[1] 1 2 3 4 5 6 7 8
```

```
b = c(4:1) # vettore di lunghezza 4
b
```

```
[1] 4 3 2 1
```

```
a - b # il vettore b è "riciclato" sul vettore a
```

```
[1] -3 -1  1  3  1  3  5  7
```

Se i vettori non hanno la stessa lunghezza (o uno non è un multiplo dell'altro) ottenete un warning

Vettori e operazioni II

Applicando una funzione a un vettore → viene applicata a **tutti** gli elementi del vettore

```
sqrt(a)
```

```
[1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.6
```

La stessa operazione si può applicare a ogni singolo elemento del vettore

```
(a - mean(a))^2 # squared deviation
```

```
[1] 12.25  6.25  2.25  0.25  0.25  2.25  6.25 12.25
```

Indicizzare i vettori

Come si va a "raggiungere" un particolare elemento all'interno del vettore?

```
nomi = c("Pasquale", "Egidio", "Debora", "Luca", "Andrea")
```

Pasquale	Egidio	Debora	Luca	Andrea
1	2	3	4	5

Indicizzare i vettori

Come si va a "raggiungere" un particolare elemento all'interno del vettore?

```
nomi = c("Pasquale", "Egidio", "Debora", "Luca", "Andrea")
```

Pasquale	Egidio	Debora	Luca	Andrea
1	2	3	4	5

```
nome_vettore[indice]
```

Indicizzare i vettori II

Pasquale	Egidio	Debora	Luca	Andrea
1	2	3	4	5

Indicizzare i vettori II

Pasquale	Egidio	Debora	Luca	Andrea
1	2	3	4	5

`nomi[1]` $\rightarrow$

Indicizzare i vettori II

Pasquale	Egidio	Debora	Luca	Andrea
1	2	3	4	5

`nomi[1]` $\rightarrow$ Pasquale

`nomi[3]` $\rightarrow$

Indicizzare i vettori II

Pasquale	Egidio	Debora	Luca	Andrea
1	2	3	4	5

`nomi[1]` $\rightarrow$ Pasquale

`nomi[3]` $\rightarrow$ Debora

`nomi[seq(2, 5, by = 2)]` $\rightarrow$

Indicizzare i vettori II

Pasquale	Egidio	Debora	Luca	Andrea
1	2	3	4	5

`nomi[1]` $\rightarrow$ Pasquale

`nomi[3]` $\rightarrow$ Debora

`nomi[seq(2, 5, by = 2)]` $\rightarrow$ Egidio, Luca

Indicizzare i vettori: Esempi

```
peso
```

```
[1]  3.0  4.5  6.0  7.5  9.0 10.5
```

```
peso[2]          # secondo elemento del vettore peso
```

```
[1] 4.5
```

```
(peso[6] = 15.2) # sostituisce il sesto elemento del v. peso
```

```
[1] 15.2
```

```
peso[seq(1, 6, by = 2)] # elementi 1, 3, 5
```

```
[1] 3 6 9
```

```
peso[2:6]        # dal 2 al 6 elemento di peso
```

```
[1]  4.5  6.0  7.5  9.0 15.2
```

```
peso[-2]       # vettore peso senza il secondo elemento
```

Indicizzare i vettori usando la logica

```
peso
```

```
[1]  3.0  4.5  6.0  7.5  9.0 15.2
```

Indicizzare i vettori usando la logica

```
peso
```

```
[1]  3.0  4.5  6.0  7.5  9.0 15.2
```

Quali sono i valori maggiori di 7?

```
peso > 7
```

```
[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

Indicizzare i vettori usando la logica

```
peso
```

```
[1]  3.0  4.5  6.0  7.5  9.0 15.2
```

Quali sono i valori maggiori di 7?

```
peso > 7
```

```
[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

Usiamo questa informazione per filtrare il nostro vettore:

```
peso[peso > 7] # valori in peso maggiori di 7
```

```
[1]  7.5  9.0 15.2
```

```
peso[peso >= 4.5 & peso < 8] # valori tra 4.5 e 8
```

```
[1] 4.5 6.0 7.5
```

Table of Contents

Un vettore che ci ha creduto abbastanza

Quel che basta per vincere una seconda dimensione

`matrix(data, nrow, ncol, byrow = TRUE)`

Crea una matrice 3×4 e la assegna all'oggetto `A`:

```
A = matrix(1:12, nrow=3, ncol = 4, byrow = FALSE)
A
```

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

WARNING: i dati all'interno della matrice devono essere tutti dello stesso tipo

Etichette

```r
rownames(A) = c(paste("riga", 1:nrow(A), sep = "_"))

colnames(A) = c(paste("colonna", 1:ncol(A), sep = "_"))

A
```

```
       colonna_1 colonna_2 colonna_3 colonna_4
riga_1         1         4         7        10
riga_2         2         5         8        11
riga_3         3         6         9        12
```

Trasposta della matrice:

```
A
```

```
       colonna_1 colonna_2 colonna_3 colonna_4
riga_1         1         4         7        10
riga_2         2         5         8        11
riga_3         3         6         9        12
```

```
t(A)
```

```
          riga_1 riga_2 riga_3
colonna_1      1      2      3
colonna_2      4      5      6
colonna_3      7      8      9
colonna_4     10     11     12
```

Creare le matrici (ancora)

Le matrici si possono anche creare concatenando vettori colonna:

```
cbind(a1 = 1:4, a2 = 5:8, a3 = 9:12)
```

```
     a1 a2 a3
[1,]  1  5  9
[2,]  2  6 10
[3,]  3  7 11
[4,]  4  8 12
```

o vettori riga:

```
rbind(a1 = 1:4, a2 = 5:8, a3 = 9:12)
```

```
   [,1] [,2] [,3] [,4]
a1    1    2    3    4
a2    5    6    7    8
a3    9   10   11   12
```

Indicizzare le matrici

Abbiamo due dimensioni:

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

```
my_matrix[righe, colonne]
```

Indicizzare le matrici: In pratica

```
A
```

```
       colonna_1 colonna_2 colonna_3 colonna_4
riga_1         1         4         7        10
riga_2         2         5         8        11
riga_3         3         6         9        12
```

`A[1, ]` →

`A[2, ]` →

`A[2, 3]` →

Indicizzare le matrici: In pratica

```
A
```

```
       colonna_1 colonna_2 colonna_3 colonna_4
riga_1         1         4         7        10
riga_2         2         5         8        11
riga_3         3         6         9        12
```

`A[1, ]` → 1, 4, 7, 10

`A[2, ]` →

`A[2, 3]` →

Indicizzare le matrici: In pratica

```
A
```

```
       colonna_1 colonna_2 colonna_3 colonna_4
riga_1         1         4         7        10
riga_2         2         5         8        11
riga_3         3         6         9        12
```

`A[1, ]` → 1, 4, 7, 10

`A[2, ]` → 2, 5, 8, 11

`A[2, 3]` →

Indicizzare le matrici: In pratica

```
A
```

```
        colonna_1 colonna_2 colonna_3 colonna_4
riga_1          1         4         7        10
riga_2          2         5         8        11
riga_3          3         6         9        12
```

`A[1, ]` $\rightarrow$ 1, 4, 7, 10

`A[2, ]` $\rightarrow$ 2, 5, 8, 11

`A[2, 3]` $\rightarrow$ 8

Table of Contents

Una matrice che ci ha creduto davvero

Davvero troppo

`array(data, c(nrow, ncol, ntab))`

Avendo 3 argomenti oltre i dati `nrow`, `ncol`, `ntab`, la loro indicizzazione prevede l'utilizzo di due virgole per accedere ai singoli argomenti:
`nome_array[righe, colonne, tab]`

Un array

```
my_array = array(1:20, c(2, 5, 3)) # 2 x 5 x 3 array
my_array
```

```
, , 1

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

, , 2

     [,1] [,2] [,3] [,4] [,5]
[1,]   11   13   15   17   19
[2,]   12   14   16   18   20

, , 3

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

Indicizzare l'array

```
my_array[1, , ]
```

```
my_array[, 2, ]
```

```
my_array[, , 3]
```

Indicizzare l'array

```
my_array[1, , ]
```

```
     [,1] [,2] [,3]
[1,]    1   11    1
[2,]    3   13    3
[3,]    5   15    5
[4,]    7   17    7
[5,]    9   19    9
```

```
my_array[, 2, ]
```

```
my_array[, , 3]
```

Indicizzare l'array

```
my_array[1, , ]
```

```
     [,1] [,2] [,3]
[1,]    1   11    1
[2,]    3   13    3
[3,]    5   15    5
[4,]    7   17    7
[5,]    9   19    9
```

```
my_array[, 2, ]
```

```
     [,1] [,2] [,3]
[1,]    3   13    3
[2,]    4   14    4
```

```
my_array[, , 3]
```

Indicizzare l'array

```
my_array[1, , ]
```

```
     [,1] [,2] [,3]
[1,]    1   11    1
[2,]    3   13    3
[3,]    5   15    5
[4,]    7   17    7
[5,]    9   19    9
```

```
my_array[, 2, ]
```

```
     [,1] [,2] [,3]
[1,]    3   13    3
[2,]    4   14    4
```

```
my_array[, , 3]
```

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

Table of Contents

Un array con più senso

Sono dei contenitori per diversi tipi di oggetti (e.g., vettori, data frames, altre liste, matrici, array ecc.)

Ai loro elementi possono essere assegnati dei nomi:

```
my_list = list(w = peso, m = mesi, s = ses1, a = A)
names(my_list)
```

```
[1] "w" "m" "s" "a"
```

```
str(my_list)
```

```
List of 4
 $ w: num [1:6] 3 4.5 6 7.5 9 15.2
 $ m: num [1:6] 5 6 8 10 12 16
 $ s: Factor w/ 3 levels "medium","high",..: 3 3 1 1 2 2
 $ a: int [1:3, 1:4] 1 2 3 4 5 6 7 8 9 10 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "riga_1" "riga_2" "riga_3"
  .. ..$ : chr [1:4] "colonna_1" "colonna_2" "colonna_3" "colonna_4
```

Indicizzare le liste

Gli elementi della lista possono essere indicizzati con $ (se la lista ha dei nomi):

```
my_list$m # vettore dei mesi
```

```
[1]  5  6  8 10 12 16
```

oppure con [[]]:

Nome dell'elemento

```
my_list[["m"]]
```

```
[1]  5  6  8 10 12 16
```

Posizione dell'elemento:

```
my_list[[2]]
```

```
[1]  5  6  8 10 12 16
```

Table of Contents

Una lista più ordinata

I data frames sono delle liste di vettori di uguale lunghezza

I diversi vettori possono contenere informazioni di diverse natura

I data frame più comuni sono i data frame in versione wide (i.e., *soggetti* $\times$ *variabili*) $\rightarrow$ `nrow(data)` $=$ numero di soggetti:

```
id = paste0("sbj", 1:6)
babies = data.frame(id, mesi, peso)
```

```
babies
```

```
    id mesi peso
1 sbj1    5  3.0
2 sbj2    6  4.5
3 sbj3    8  6.0
4 sbj4   10  7.5
5 sbj5   12  9.0
6 sbj6   16 15.2
```

Indicizzare i data frame

Vale tutto quello visto per le matrici:

Prima riga del data frame `babies`

```
babies[1, ]
```

Prima colonna del data frame

```
babies
babies[, 1]
```

In più:

```
babies$mesi # colonna mesi di babies

babies$mesi[2] # secondo elemento del vettore colonna

babies[, "id"] # column id

babies[2, ] # second row of babies (obs on baby 2)
```

Logic applies:

```
babies[babies$peso > 7, ] # filtra per tutte le righe con pes
```

Working with data frames II

```
dim(babies) # data frame con 6 righe e 3 colonne
```

```
[1] 6 3
```

```
names(babies) # = colnames(babies)
```

```
[1] "id"   "mesi" "peso"
```

```
View(babies) # open data viewer
```

Questi comandi possono essere usati anche su altri oggetti `R`