Vettori, Matrici e Dataframe

Test per le organizzazioni

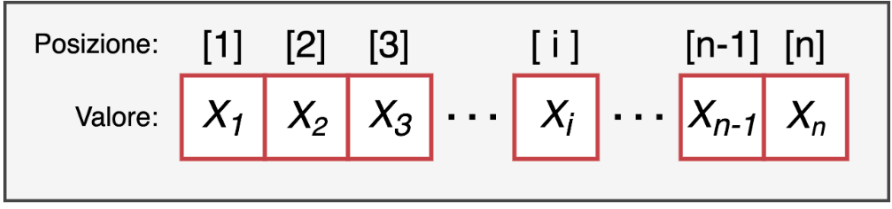
Ottavia M. Epifania
ottavia.epifania@unipd.it
Margherita Calderan
margherita.calderan@unipd.it

Università di Padova

03/2026

Vettori

I vettori sono una struttura dati unidimensionale e sono la più semplice presente in R.

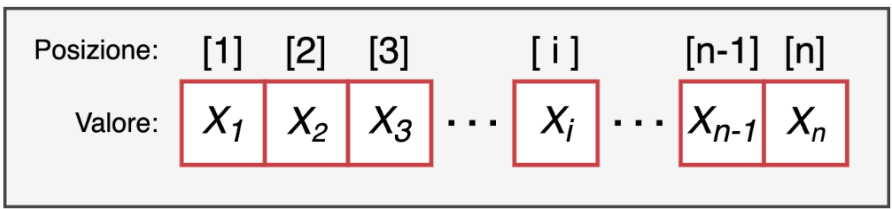


Caratteristiche di un vettore

- **la lunghezza**: il numero di elementi da cui è formato il vettore
- **la tipologia**: la tipologia di dati da cui è formato il vettore. Un vettore infatti deve esssere formato da **elementi tutti dello stesso tipo**!

Caratteristiche degli elementi di un vettore

- **un valore**: il valore dell'elemento che può essere di qualsiasi tipo ad esempio un numero o una serie di caratteri
- **un indice di posizione**: un numero intero positivo che identifica la sua posizione all'interno del vettore.



Creare un vettore

I vettori si possono creare attraverso il comando `c()`, indicando tra le parentesi i valori degli elementi nella sucessione desiderata e separati da una virgola.

```
num_vect = c(1,2,3,4)
```

```
char_vect = c("R","R","R","ok")
```

Attraverso il comando `length()` è possibile ottenere la lunghezza del vettore

```
length(num_vect)
```

```
[1] 4
```

```
length(char_vect)
```

```
[1] 4
```

Tiplogia di vettore

La tipologia di dati da cui è formato il vettore.

```
class(num_vect)
```

```
[1] "numeric"
```

```
class(char_vect)
```

```
[1] "character"
```

Un vettore deve esssere formato da **elementi tutti dello stesso tipo**!

```
wrong = c(1,2,3,"non so", 4)
class(wrong)
```

```
[1] "character"
```

```
wrong
```

```
[1] "1"      "2"      "3"      "non so" "4"
```

Altrimenti si “rischia” che tutto venga trasformato a carattere.

```
correct = c(1,2,3,NA, 4)
class(correct)
```

```
[1] "numeric"
```

```
correct
```

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

is.* & as.*

Possiamo testare o convertire (quando possibile) la tipologia del vettore attraverso queste funzioni `is.` & `as.`

Vettore di tipo `character`

```
char_vect
```

```
[1] "R"  "R"  "R"  "ok"
```

```
is.character(char_vect)
```

```
[1] TRUE
```

```
as.numeric(char_vect) #!!
```

```
[1] NA NA NA NA
```

is.* & as.*

Vettore di tipo numeric

```
num_vect
```

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

```
is.numeric(num_vect)
```

```
[1] TRUE
```

```
as.character(num_vect) #!!
```

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

is.* & as.*

Vettore di tipo `logical`

```
logi_vect = c(TRUE,FALSE,TRUE)
is.logical(logi_vect)
```

```
[1] TRUE
```

```
as.numeric(logi_vect)
```

```
[1] 1 0 1
```

Indicizzazione

Possiamo selezionare, eliminare, estrarre elementi semplicemente usando l'indice di posizione tramite le parentesi quadre vettore [pos] .

```
# Creo un vettore formato da 10 numeri casuali
my_vect = round(runif(n = 10,min = 1, max = 100))
my_vect
```

```
 [1] 24 32 14 36 55  9 93 18 22 74
```

```
my_vect[1] # estraggo il primo elemento
```

```
[1] 24
```

```
my_vect[1:5] # estraggo i primi 5 elementi
```

```
[1] 24 32 14 36 55
```

```
my_vect[c(1,4,2,9)] # estraggo elementi a scelta
```

```
[1] 24 36 32 22
```

Indicizzazione Logica

Possiamo selezionare elementi dal vettore basandoci su specifiche condizioni logiche: **TRUE** e **FALSE**.

```
(numeri = 1:7)
```

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

```
numeri>2 & numeri<5
```

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

```
numeri[numeri>2 & numeri<5]
```

```
[1] 3 4
```

Operazioni matematiche sui vettori

Possiamo eseguire operazioni sui vettori, ed applicare la stessa operazione a tutti gli elementi del vettore (element-wise)

```
# ?rep
new_vect = rep(2:4, each = 2)
new_vect
```

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

```
# potete svolgere qualsiasi operazione
new_vect/2
```

```
[1] 1.0 1.0 1.5 1.5 2.0 2.0
```

Fattori

I fattori sono una tipologia di dato peculiare, ma simile a semplici vettori `character`

Creare un fattore: `factor()`

Attraverso la funzione `factor()` è possibile creare un fattore, specificando: il vettore character, l'ordine dei livelli del fattore (levels =), le etichette per ciascun livello (labels =)

```
my_fact1 = factor(x = c("LT", "LM", "PhD", "LT", "LM", "PhD"),
                  levels = c("LT", "LM", "PhD"),
                    labels = c("LT", "LM", "PhD"))
my_fact1
```

```
[1] LT  LM  PhD LT  LM  PhD
Levels: LT LM PhD
```

Creare un fattore: `as.factor()`

La funzione `rep()` permette di ripetere gli elementi di un vettore. In questo esempio creo prima un vettore di tipo `character` e poi lo trasformo a fattore attraverso il comando `as.factor()`.

```
# come funziona rep?
rep(c("LT", "LM", "PhD"), times = 2)
```

```
[1] "LT"  "LM"  "PhD" "LT"  "LM"  "PhD"
```

```
# creo il vettore
char_vect = rep(c("LT", "LM", "PhD"), times = 2)

# Trasformo il vettore character in fattore
(my_fact2 = as.factor(char_vect)) # cosa notate di diverso ?
```

```
[1] LT  LM  PhD LT  LM  PhD
Levels: LM LT PhD
```

Fattori e Ordine Alfabetico

Perché l'ordine dei livelli differisce? Di default, la funzione `factor()` assegna l'ordine dei livelli alfabeticamente, a meno che non lo forziamo con l'argomento `levels`.

```
str(my_fact2)
```

```
 Factor w/ 3 levels "LM","LT","PhD": 2 1 3 2 1 3
```

```
str(my_fact1)
```

```
 Factor w/ 3 levels "LT","LM","PhD": 1 2 3 1 2 3
```

I fattori sono trattati come numeri interi. R prende i valori unici, li mette in ordine alfabetico e assegna loro progressivamente i numeri 1, 2, 3, ecc. Questo rispecchia come R valuta le stringhe di testo:

```
"a" > "b"
```

```
[1] FALSE
```

```
"abba" > "aac"
```

```
[1] TRUE
```

I livelli

I fattori permettono di avere dei livelli **`levels()`** come metadati,

```
levels(my_fact1)
```

```
[1] "LT"  "LM"  "PhD"
```

... a prescindere da quali dati siano effettivamente presenti nel vettore.

Per esempio se creo un fattore composto solo dagli elementi di my_fact1 diversi da "LM":

```
my_fact1 # fattore
```

```
[1] LT  LM  PhD LT  LM  PhD
Levels: LT LM PhD
```

```
my_fact1!="LM" # condizione da valutare per tutta la lunghezza del vettore
```

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

```
my_fact1[my_fact1!="LM"] # prendo solo gli elementi diversi da "LM"
```

```
[1] LT  PhD LT  PhD
Levels: LT LM PhD
```

```
my_fact3 = my_fact1[my_fact1!="LM"] #creo l'oggetto my_fact3
```

I livelli di my_fact2 saranno gli stessi di my_fact1 ("LM" incluso) anche se "LM" non è presente come osservazione

```
my_fact3
```

```
[1] LT  PhD LT  PhD
Levels: LT LM PhD
```

E' possibile però eslcudere i livelli non più utili attraverso il comando `droplevels()`:

```
# come sarebbe my_fact3?
droplevels(my_fact3)
```

```
[1] LT  PhD LT  PhD
Levels: LT PhD
```

```
# modifico my_fact2 eliminando i livelli inutili
my_fact3 = droplevels(my_fact3)
```

Ora facciamo un po' di pratica!

Aprite e tenete aperto questo link:

https://etherpad.wikimedia.org/p/test-organizzazioni

1. Vettori
2. Fattori
3. Matrici
   - Caratteristiche
   - Indicizzazione
   - Vettori e Matrici
   - Operazioni con le matrici
4. Dataframe

Matrici

Le matrici sono una struttura dati **bidimensionale** (caratterizzate da 2 dimensioni `dim()`) dove il numero di righe rappresenta la dimensione 1 e il numero di colonne la dimensione 2.

```
my_mat = matrix(data = 1:10, nrow = 2, ncol = 5)
my_mat
```

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

Matrici

```
my_mat = matrix(data = 1:10, nrow = 2,
                ncol = 5)
nrow(my_mat)
```

```
[1] 2
```

```
ncol(my_mat)
```

```
[1] 5
```

Caratteristiche

- Possono contenere **una sola tipologia** di dati
- Essendo **bidimensionali**, abbiamo bisogno di due indici di posizione (righe e colonne) per identificare un elemento
- Possono essere viste come un **insieme** di singoli **vettori**

Caratteristiche

Il numero di righe e colonne non deve essere lo stesso necessariamente (matrice quadrata) ma il numero di righe deve essere compatibile con il vettore data:

```
matrix(data = 1:10, ncol = 3, nrow = 3)
```

```
Warning in matrix(data = 1:10, ncol = 3, nrow = 3): data length [10] is not a
sub-multiple or multiple of the number of rows [3]
```

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

Cosa fa R di default?

```
matrix(data = 1:10, ncol = 3, nrow = 3)
```

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

```
matrix(data = 1:2, ncol = 3, nrow = 3)
```

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

warnings: la funzione ci informa di qualcosa di potenzialmente problematico, ma (circa!!) tutto liscio

Indicizzazione

Per identificare uno o più elementi nella matrice abbiamo bisogno di indici/e di riga e/o colonna separati da virgola, sempre con le parentesi quadre: **matrice[riga, colonna]**

```
my_mat
```

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

```
my_mat[1,1]
```

```
[1] 1
```

E' possibile anche selezionare un'intera riga o colonna

```
my_mat[1,]
```

```
[1] 1 3 5 7 9
```

```
my_mat[,1]
```

```
[1] 1 2
```

Vettori e Matrici

I vettori si creano attraverso la funzione `c()` e possono essere concatenati tra loro sempre attraverso la stessa funzione:

```
my_vect1 = c(1:4)
my_vect2 = c(5:10)

my_vect12 = c(my_vect1,my_vect2)
my_vect12
```

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

```
my_mat1 = matrix(data = 1:4,nrow = 2, ncol = 2)
my_mat1
```

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

```
my_mat2 = matrix(data = 5:8,nrow = 2, ncol = 2)
my_mat2
```

```
     [,1] [,2]
[1,]    5    7
[2,]    6    8
```

Le matrici possono essere unite tra loro attraverso i comandi:

cbind()

```
cbind(my_mat1, my_mat2)
```

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

rowbind()

```
rbind(my_mat1, my_mat2)
```

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

Operazioni con le matrici

Come per i vettori, anche alle matrici si possono applicare operazioni matematiche:

```
my_mat = matrix(data = 1:4,nrow = 2, ncol = 2)

# elemet-wise
my_mat*my_mat
```

```
     [,1] [,2]
[1,]    1    9
[2,]    4   16
```

Come per i vettori, anche alle matrici si possono applicare operazioni matematiche:

```
# Prodotto matriciale
my_mat%*%my_mat
```

```
     [,1] [,2]
[1,]    7   15
[2,]   10   22
```

```
# (1*1 + 3*2) , (1*3 + 3*4)
# (2*1 + 4*2) , (2*3 + 4*4)
```

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

- Creazione
- Attributi
- Indicizzazione
- Combinare Dataframe
- Esportazione e importazione dati

Dataframe

Il dataframe è la struttura più "complessa", utile e potente di R.

- ogni elemento è un **vettore** con un **nome associato** (aka una colonna)
- ogni colonna deve avere lo stesso numero di elementi
- di conseguenza ogni riga ha lo stesso numero di elementi (**struttura rettangolare**)

Creazione

Si creano attraverso il comando `data.frame`

```
# Creo un dataframe con 3 colonne
my_df = data.frame( numeri = 1:4, lettere = letters[1:4],
                    normale = rnorm(n = 4, mean = 0, sd = 1))
my_df
```

```
  numeri lettere     normale
1      1       a  1.11711371
2      2       b -0.35321441
3      3       c -0.60047027
4      4       d -0.07319191
```

Attributi

Possiamo utlizzare le funzioni **names()** , **dim()**, **nrow()**, **ncol()**... per ottenere informazioni sulle caratteristiche del dataframe.

La funzione più utile è **str()** poichè ci restituisce una veloce overview della struttura del dataframe: dimensioni, tipi di variabili,...

```
str(my_df)
```

```
'data.frame':   4 obs. of  3 variables:
 $ numeri : int  1 2 3 4
 $ lettere: chr  "a" "b" "c" "d"
 $ normale: num  1.1171 -0.3532 -0.6005 -0.0732
```

Indicizzazione

```
my_df[1]  # estraggo un data.frame 5x1
```

```
  numeri
1      1
2      2
3      3
4      4
```

```
my_df[[1]] # estraggo la prima colonna del data.frame
```

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

```
my_df[1,1] #  estraggo il primo elemento della prima colonna del data.frame
```

```
[1] 1
```

Indicizzazione $

Estraggo la prima colonna del data.frame:

```
my_df$numeri
```

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

...estraggo il primo elemento della prima colonna del data.frame:

```
my_df$numeri[1]
```

```
[1] 1
```

Indicizzazione Logica

Una delle operazioni più comuni che dovrete affrontare sarà sicuramente quella di estrarre/valutare un sottoinsieme di valori presenti nel vostro dataset

Includo solo le righe per cui alla colonna numeri i valori sono maggiori di 2

```
my_df[my_df$numeri > 2, ]
```

```
  numeri lettere     normale
3      3       c -0.60047027
4      4       d -0.07319191
```

```
my_df[my_df[1] > 2, ]
```

```
  numeri lettere     normale
3      3       c -0.60047027
4      4       d -0.07319191
```

```
my_df
```

```
  numeri lettere     normale
1      1       a  1.11711371
2      2       b -0.35321441
3      3       c -0.60047027
4      4       d -0.07319191
```

```
my_df[my_df$numeri > 2 & my_df$numeri < 4, ]
```

```
my_df
```

```
  numeri lettere     normale
1      1       a  1.11711371
2      2       b -0.35321441
3      3       c -0.60047027
4      4       d -0.07319191
```

```
my_df[my_df$numeri > 2 & my_df$numeri < 4, ]
```

```
  numeri lettere    normale
3      3       c -0.6004703
```

```
my_df
```

```
  numeri lettere     normale
1      1       a  1.11711371
2      2       b -0.35321441
3      3       c -0.60047027
4      4       d -0.07319191
```

```
my_df[my_df$numeri== 2,  2]
```

```
my_df
```

```
  numeri lettere     normale
1      1       a  1.11711371
2      2       b -0.35321441
3      3       c -0.60047027
4      4       d -0.07319191
```

```
my_df[my_df$numeri== 2,  2]
```

```
[1] "b"
```

```
my_df
```

```
  numeri lettere     normale
1      1       a  1.11711371
2      2       b -0.35321441
3      3       c -0.60047027
4      4       d -0.07319191
```

```
my_df[my_df$numeri== 2,  "lettere"]
```

```
my_df
```

```
  numeri lettere     normale
1      1       a  1.11711371
2      2       b -0.35321441
3      3       c -0.60047027
4      4       d -0.07319191
```

```
my_df[my_df$numeri== 2,  "lettere"]
```

```
[1] "b"
```

Indicizzazione subset()

```
my_df = data.frame(numeri = rep(1:3,each = 3),
                   lettere = rep(letters[1:3],3),
                   normale = rnorm(n = 9, mean = 0, sd = 1))

#visualizzo le prime 5 righe attraverso il comando head
head(my_df, n = 5)
```

```
  numeri lettere     normale
1      1       a -0.81363812
2      1       b  0.02063969
3      1       c  0.48064740
4      2       a  2.17908197
5      2       b  0.81318973
```

Attraverso la funzione `subset()` è possibile ottenere un dataframe formato da un sottoinsieme di elementi.

```
subset(my_df, subset = lettere == "a" & numeri > 2)
```

```
  numeri lettere    normale
7      3       a -0.5727113
```

L'argomento `subset=` definisce la condizione da valutare per la selezione degli elementi di `my_df`.

```r
subset(my_df, subset = lettere == "a" & numeri > 2)
```

```
  numeri lettere    normale
7      3       a -0.5727113
```

è equivalente a:

```r
my_df[my_df$lettere == "a" & my_df$numeri > 2,]
```

```
  numeri lettere    normale
7      3       a -0.5727113
```

`subset(df, select = ...)`

E' possibile anche selezionare specigiche colonne attraverso l'argomento ***select***:

```
subset(my_df, select = c(lettere, numeri))
```

```
#visualizzo le prime tre righe attraverso il comando head
head(subset(my_df, select = c(lettere, numeri)), n = 3)
```

```
  lettere numeri
1       a      1
2       b      1
3       c      1
```

`subset(df, subset = ..., select = ...)`

Possiamo anche combinare le due cose:

```
subset(my_df, subset = lettere == "a" & numeri > 2,
       select = c(lettere, numeri))
```

```
  lettere numeri
7       a      3
```

- valuto la condizione per riga (`subset = lettere == "a" & numeri > 2`)
- seleziono le colonne (`subset = c(lettere, numeri)`)

La maggiorparte delle volte vi troverete ad accedere alle variabili tramite l'operatore $. Questo comando può essere utilizzato anche per creare una nuova variabile...

```
my_df$numeri
```

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

```
# creo una variablie che è la somma di numeri e normale
my_df$somma = my_df$numeri + my_df$normale
str(my_df)
```

```
'data.frame':   9 obs. of  4 variables:
 $ numeri : int  1 1 1 2 2 2 3 3 3
 $ lettere: chr  "a" "b" "c" "a" ...
 $ normale: num  -0.8136 0.0206 0.4806 2.1791 0.8132 ...
 $ somma  : num  0.186 1.021 1.481 4.179 2.813 ...
```

Potete sia creare che modificare variabili accedendo attraverso $

```
# Modfico la variabile num aggiungendo 1
my_df$numeri = my_df$numeri+1

# Creo una terza variabile composta dalla varibile num e let
my_df$both = paste(my_df$numeri,my_df$lettere, sep = "_") # ?paste

str(my_df)
```

```
'data.frame':   9 obs. of  5 variables:
 $ numeri : num  2 2 2 3 3 3 4 4 4
 $ lettere: chr  "a" "b" "c" "a" ...
 $ normale: num  -0.8136 0.0206 0.4806 2.1791 0.8132 ...
 $ somma  : num  0.186 1.021 1.481 4.179 2.813 ...
 $ both   : chr  "2_a" "2_b" "2_c" "3_a" ...
```

Combinare Dataframe

Essendo simili a delle matrici, i dataframe si possono combinare tra loro attraverso le funzioni **rbind()**:

```
my_df2 = data.frame(numero = 1:9, lettere = letters[1:9],
                    normale = rnorm(9,mean = 0,sd = 1),
                    somma = my_df$somma,
                    both = paste(1:9,letters[1:9], sep = "_"))
str(my_df2)
```

```
'data.frame':   9 obs. of  5 variables:
 $ numero : int  1 2 3 4 5 6 7 8 9
 $ lettere: chr  "a" "b" "c" "d" ...
 $ normale: num  1.351 -0.779 0.72 0.359 -0.375 ...
 $ somma  : num  0.186 1.021 1.481 4.179 2.813 ...
 $ both   : chr  "1_a" "2_b" "3_c" "4_d" ...
```

Unisco i due Dataframe

```
my_df3 = rbind(my_df,my_df2)
```

```
Error in `match.names()`:
! names do not match previous names
```

```
str(my_df)
```

```
'data.frame':   9 obs. of  5 variables:
 $ numeri : num  2 2 2 3 3 3 4 4 4
 $ lettere: chr  "a" "b" "c" "a" ...
 $ normale: num  -0.8136 0.0206 0.4806 2.1791 0.8132 ...
 $ somma  : num  0.186 1.021 1.481 4.179 2.813 ...
 $ both   : chr  "2_a" "2_b" "2_c" "3_a" ...
```

```
str(my_df2)
```

```
'data.frame':   9 obs. of  5 variables:
 $ numero : int  1 2 3 4 5 6 7 8 9
 $ lettere: chr  "a" "b" "c" "d" ...
 $ normale: num  1.351 -0.779 0.72 0.359 -0.375 ...
 $ somma  : num  0.186 1.021 1.481 4.179 2.813 ...
 $ both   : chr  "1_a" "2_b" "3_c" "4_d" ...
```

Unisco i due Dataframe

- I Dataframe devono avere lo stesso numero di colonne
- I nomi delle colonne devono essere identici

Sistemo i nomi

```
names(my_df2)
```

```
[1] "numero"  "lettere" "normale" "somma"   "both"
```

```
names(my_df)
```

```
[1] "numeri"  "lettere" "normale" "somma"   "both"
```

```
# voglio che i names di my_df2 corrispondano ai names di my_df
names(my_df2) = names(my_df)
```

Unisco i dataframe

```
my_df3 = rbind(my_df,my_df2)
str(my_df3)
```

```
'data.frame':   18 obs. of  5 variables:
 $ numeri : num  2 2 2 3 3 3 4 4 4 1 ...
 $ lettere: chr  "a" "b" "c" "a" ...
 $ normale: num  -0.8136 0.0206 0.4806 2.1791 0.8132 ...
 $ somma  : num  0.186 1.021 1.481 4.179 2.813 ...
 $ both   : chr  "2_a" "2_b" "2_c" "3_a" ...
```

Potrebbe anche capitarvi di dover raccogliere differenti tipi di dato dallo stesso partecipante, e successivamente combinare le informazioni raccolte...

Dataframe contente i tempi di reazione:

```
df_rt = data.frame(subj = factor(rep(c("caio","tizio"),each = 400)),
                   cond = factor(rep(c("easy","hard"),
                                     each = 200, times = 2)),
                     rt = c(rlnorm(n = 400, meanlog = -1, sdlog = .25),
                            rlnorm(n = 400, meanlog = -.7, sdlog =.3)))
str(df_rt)
```

```
'data.frame':   800 obs. of  3 variables:
 $ subj: Factor w/ 2 levels "caio","tizio": 1 1 1 1 1 1 1 1 1 1 ...
 $ cond: Factor w/ 2 levels "easy","hard": 1 1 1 1 1 1 1 1 1 1 ...
 $ rt  : num  0.519 0.519 0.421 0.404 0.266 ...
```

Dataframe contente l'età:

```
df_age = data.frame(subj = factor(c("caio","tizio")), age = c(20,3))
str(df_age)
```

```
'data.frame':   2 obs. of  2 variables:
 $ subj: Factor w/ 2 levels "caio","tizio": 1 2
 $ age : num  20 3
```

In questo caso, è possibile utilizzare la funzione **merge()**:

```
df_all_1 = merge(x = df_rt, y = df_age, by="subj")
str(df_all_1)
```

```
'data.frame':   800 obs. of  4 variables:
 $ subj: Factor w/ 2 levels "caio","tizio": 1 1 1 1 1 1 1 1 1 1 ...
 $ cond: Factor w/ 2 levels "easy","hard": 1 1 1 1 1 1 1 1 1 1 ...
 $ rt  : num  0.519 0.519 0.421 0.404 0.266 ...
 $ age : num  20 20 20 20 20 20 20 20 20 20 ...
```

Esportazione e importazione dati

In R è possibile importare dati in molti formati differenti, più comunemente vi troverete ad importare dati ***.csv*** oppure ***.xlsx***.

Qui per esempio, esporto i dataframe in tre formati differenti...

```
library(readr) # carico il pacchetto readr
library(writexl) # carico il pacchetto writexl

write.csv(df_rt, file = "data/df_rt.csv",row.names = FALSE)

save(df_rt, file = "data/df_rt.rda") # formato R

write_xlsx(df_age, path = "data/df_age.xlsx")
```

Importo

```
df_rt_impo = read_csv("data/df_rt.csv") #utilizza il pacchetto readr

load(file ="data/df_rt.rda") # formato R

library(readxl) # carico il pacchetto readxl
df_age_impo = read_xlsx("dat/df_age.xlsx")
```

Ora facciamo un po' di pratica!

Aprite e tenete aperto questo link:

https://etherpad.wikimedia.org/p/test-organizzazioni