

Vettori, Matrici e Dataframe

Test per le organizzazioni

Ottavia M. Epifania

ottavia.epifania@unipd.it

Margherita Calderan

margherita.calderan@unipd.it

Università di Padova

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

1 Vettori

- Indicizzazione

2 Fattori

3 Matrici

4 Dataframe

Vettori

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

Posizione:	[1]	[2]	[3]	...	[i]	...	[n-1]	[n]
Valore:	x_1	x_2	x_3	...	x_i	...	x_{n-1}	x_n

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 essere 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.

Posizione:	[1]	[2]	[3]	...	[i]	...	[n-1]	[n]
Valore:	X_1	X_2	X_3	...	X_i	...	X_{n-1}	X_n

Creare un vettore

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

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

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

Tipologia di vettore

La tipologia di dati da cui è formato il vettore.

```
class(num_vect)
```

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

```
class(char_vect)
```

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

Tipologia di vettore

Un vettore deve essere 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)
```

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

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

Indicizzazione

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] 38 83 82 93 15 71 35 81 31 87
```

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

```
[1] 38
```

Indicizzazione

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

```
[1] 38 83 82 93 15
```

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

```
[1] 38 93 83 31
```

Indicizzazione

Indicizzazione Negativa

Allo stesso modo possiamo decidere di estrarre tutti gli elementi del vettore eccetto alcuni

```
my_vect[-c(1)] #tutti tranne il primo elemento
```

```
[1] 83 82 93 15 71 35 81 31 87
```

```
my_vect[-c(1:9)] #tutti tranne i primi 10
```

```
[1] 87
```

Indicizzazione

Indicizzazione Logica

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

```
numeri = 1:7; numeri
```

```
[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
```

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

Fattori

I fattori sono una tipologia di dato peculiare e per quanto simile a semplici **characters** in realtà sono un tipo di vettore **integer** con delle proprietà aggiuntive.

Creare un fattore: `as.factor`

```
char_vect = rep(c("hello", "ciao", "hola"), each = 2)  
char_vect
```

```
[1] "hello" "hello" "ciao"   "ciao"   "hola"   "hola"
```

```
my_fact = as.factor(char_vect)  
my_fact
```

```
[1] hello hello ciao  ciao  hola  hola  
Levels: ciao hello hola
```

Creare un fattore: factor()

```
my_fact = factor(x = rep(c("hello", "ciao", "hola"), each = 2),  
                 levels = c("hello", "ciao", "hola"),  
                 labels = c("hello", "ciao", "hola"))
```

```
my_fact
```

```
[1] hello hello ciao  ciao  hola  hola  
Levels: hello ciao hola
```

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

```
levels(my_fact)
```

```
[1] "hello" "ciao"  "hola"
```

A prescindere da quali siano effettivamente presenti nel vettore. Per esempio se creo un fattore composto solo dagli elementi di my_fact diversi da ciao:

```
my_fact2 = my_fact[my_fact!="ciao"]
```

I livelli di my_fact2 saranno gli stessi di my_fact (“ciao” incluso) anche se “ciao” non è presente come osservazione:

```
my_fact2
```

```
[1] hello hello hola  hola  
Levels: hello ciao hola
```

```
levels(my_fact2)
```

```
[1] "hello" "ciao"  "hola"
```

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

```
# come sarebbe my_fact2?  
droplevels(my_fact2)
```

```
[1] hello hello hola  hola  
Levels: hello hola
```

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

```
[1] hello hello hola  hola  
Levels: hello hola
```

Ora facciamo un po' di pratica!

Aprite e tenete aperto questo link:

<https://etherpad.wikimedia.org/p/arca-corsoR>

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

1 Vettori

2 Fattori

3 Matrici

- Matrici - Indicizzazione

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
```

Matrici - 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**

Matrici - 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

Matrici - Indicizzazione

Matrici - 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
```

Matrici - Indicizzazione

E' possibile anche selezionare un'intera riga o colonna

```
my_mat[1,]
```

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

```
my_mat[,1]
```

```
[1] 1 2
```

Matrici - Indicizzazione

Indicizzazione logica

```
my_mat>2
```

```
      [,1] [,2] [,3] [,4] [,5]
[1,] FALSE TRUE TRUE TRUE TRUE
[2,] FALSE TRUE TRUE TRUE TRUE
```

```
my_mat
```

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

Tutti gli elementi maggiori di due

```
my_mat[my_mat>2]
```

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

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
```

Matrici - Indicizzazione

```
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
```

Matrici - Indicizzazione

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
```

Matrici - Indicizzazione

`rowbind()`

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

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

Cosa notate di “strano”?

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)
```

element-wise

```
my_mat*my_mat
```

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

Matrici - Indicizzazione

Operazioni con le matrici

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)
```

Matrici - Indicizzazione

Ora facciamo un po' di pratica!

Aprite e tenete aperto questo link:

<https://etherpad.wikimedia.org/p/arca-corsoR>

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

- Creazione
- Indicizzazione
- Combinare Dataframes
- 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	-0.9032127
2	2	b	1.3175695
3	3	c	0.7610726
4	4	d	-0.9650070

Attributi

Possiamo utilizzare 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 -0.903 1.318 0.761 -0.965
```

Indicizzazione

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

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

Indicizzazione - Operatori relazionali

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

Indicizzazione

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  0.7610726
4      4         d -0.9650070
```

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

```
  numeri lettere    normale
3      3         c  0.7610726
4      4         d -0.9650070
```

Individuazione

```
my_df
```

	numeri	lettere	normale
1	1	a	-0.9032127
2	2	b	1.3175695
3	3	c	0.7610726
4	4	d	-0.9650070

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

Individuazione

```
my_df
```

	numeri	lettere	normale
1	1	a	-0.9032127
2	2	b	1.3175695
3	3	c	0.7610726
4	4	d	-0.9650070

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

	numeri	lettere	normale
3	3	c	0.7610726

Indicizzazione

```
my_df
```

```
    numeri lettere      normale
1       1      a -0.9032127
2       2      b  1.3175695
3       3      c  0.7610726
4       4      d -0.9650070
```

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

Indicizzazione

```
my_df
```

	numeri	lettere	normale
1	1	a	-0.9032127
2	2	b	1.3175695
3	3	c	0.7610726
4	4	d	-0.9650070

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

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

Indicizzazione

```
my_df
```

```
    numeri lettere      normale
1       1      a -0.9032127
2       2      b  1.3175695
3       3      c  0.7610726
4       4      d -0.9650070
```

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

Indicizzazione

```
my_df
```

	numeri	lettere	normale
1	1	a	-0.9032127
2	2	b	1.3175695
3	3	c	0.7610726
4	4	d	-0.9650070

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

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

Indicizzazione

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	0.19451687
2	1	b	-2.68863174
3	1	c	0.01095729
4	2	a	-0.04894002
5	2	b	0.59013812

Indicizzazione

Indicizzazione subset()

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

	numeri	lettere	normale
7	3	a	-1.928077

Individuazione

Equivalent to:

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

	numeri	lettere	normale
7	3	a	-1.928077

Indicizzazione

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

E' possibile anche selezionare colonne piuttosto che righe 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

Indicizzazione

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

Possiamo anche combinare le due cose:

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

```
head(subset(my_df, subset = lettere == "a" & numeri > 2,  
            select = c(lettere, numeri)), n = 3)
```

	lettere	numeri
7	a	3

Indicizzazione

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

```
# creo una variabile 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 0.1945 -2.6886 0.011 -0.0489 0.5901 ...
 $ somma  : num 1.19 -1.69 1.01 1.95 2.59 ...
```

Indicizzazione

Potete sia creare che modificare variabili accedendo attraverso \$

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

# Creo una terza variabile composta dalla variabile 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  0.1945 -2.6886 0.011 -0.0489 0.5901 ...
 $ somma  : num  1.19 -1.69 1.01 1.95 2.59 ...
 $ both    : chr  "2_a" "2_b" "2_c" "3_a" ...
```

Combinare Dataframes

Combinare Dataframes

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 0.9362 0.5735 -1.5671 -2.0397 -0.0408 ...  
 $ somma : num 1.19 -1.69 1.01 1.95 2.59 ...  
 $ both   : chr "1_a" "2_b" "3_c" "4_d" ...
```

Combinare Dataframes

Unisco i due dataframes

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

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

Combinare Dataframes

```
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 0.1945 -2.6886 0.011 -0.0489 0.5901 ...  
$ somma : num 1.19 -1.69 1.01 1.95 2.59 ...  
$ 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 0.9362 0.5735 -1.5671 -2.0397 -0.0408 ...  
$ somma : num 1.19 -1.69 1.01 1.95 2.59 ...  
$ both   : chr "1_a" "2_b" "3_c" "4_d" ...
```

Combinare Dataframes

Unisco i due dataframes

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

Combinare Dataframes

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)
```

Combinare Dataframes

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 0.1945 -2.6886 0.011 -0.0489 0.5901 ...  
$ somma : num 1.19 -1.69 1.01 1.95 2.59 ...  
$ both   : chr "2_a" "2_b" "2_c" "3_a" ...
```

Combinare Dataframes

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

Combinare Dataframes

```
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 0.361 0.417 0.36 0.487 0.369 ...
```

Combinare Dataframes

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
```

Combinare Dataframes

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 0.361 0.417 0.36 0.487 0.369 ...
$ age : num 20 20 20 20 20 20 20 20 20 20 ...
```

Combinare Dataframes

oppure la funzione `_join()`:

```
library(dplyr) # carico il pacchetto dplyr

df_all_2 = left_join(x = df_rt, y = df_age, by = c("subj"))
str(df_all_2)

'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 0.361 0.417 0.36 0.487 0.369 ...
 $ age : num 20 20 20 20 20 20 20 20 20 20 ...
```

Esportazione e importazione dati

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")
```

Esportazione e importazione dati

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")
```

Esportazione e importazione dati

Formato R

E' consigliato avere una versione del dataset in formato .csv e un'altra in formato .rda (formato specifico di R)

Esportazione e importazione dati

Controllo la struttura ed il tipo di dati

```
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  0.361 0.417 0.36 0.487 0.369 ...
```

Controllo la struttura ed il tipo di dati

```
str(rt_csv)
```

```
spc_tbl_ [800 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
$ subj: chr [1:800] "caio" "caio" "caio" "caio" ...
$ cond: chr [1:800] "easy" "easy" "easy" "easy" ...
$ rt  : num [1:800] 0.361 0.417 0.36 0.487 0.369 ...
- attr(*, "spec")=
.. cols(
..   subj = col_character(),
..   cond = col_character(),
..   rt = col_double()
.. )
- attr(*, "problems")=<externalptr>
```

Modifico il tipo di dato

```
rt_csv$subj = as.factor(rt_csv$subj)
rt_csv$cond = as.factor(rt_csv$cond)
str(rt_csv)
```

```
spc_tbl_ [800 x 3] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
$ 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 [1:800] 0.361 0.417 0.36 0.487 0.369 ...
- attr(*, "spec")=
.. cols(
..   subj = col_character(),
..   cond = col_character(),
..   rt = col_double()
.. )
- attr(*, "problems")=<externalptr>
```

Esportazione e importazione dati

Ora facciamo un po' di pratica!

Aprite e tenete aperto questo link:

<https://etherpad.wikimedia.org/p/arca-corsoR>