Vettori, Matrici e Dataframe

Test per le organizzaioni

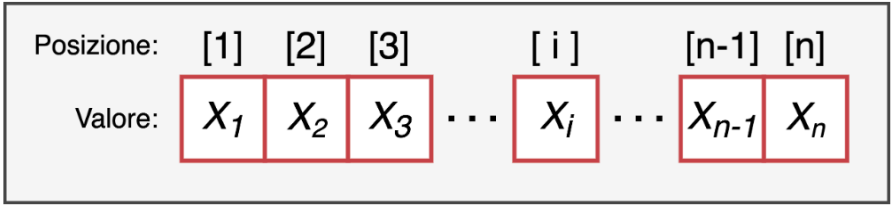
Ottavia M. Epifania
ottavia.epifania@unipd.it

Margherita Calderan
margherita.calderan@unipd.it

Università di Padova

Vettori

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

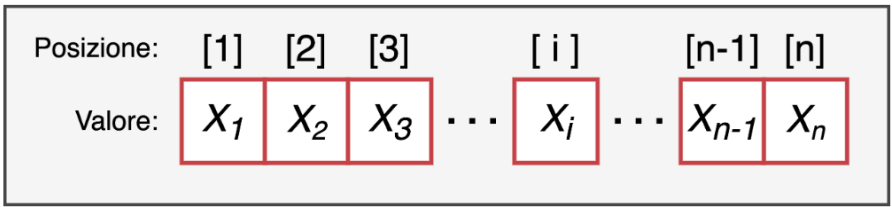


Caratteristiche di un vettore

- **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")
```

Tiplogia di vettore

La tipologia di dati da cui è formato il vettore.

```
class(num_vect)
```

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

```
class(char_vect)
```

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

Tiplogia di vettore

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"
```

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

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] 13 12 28 68 86 69 95 12 19 66
```

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

```
[1] 13
```

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

```
[1] 13 12 28 68 86
```

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

```
[1] 13 68 12 19
```

Inidicizzazione 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] 12 28 68 86 69 95 12 19 66
```

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

```
[1] 66
```

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
```

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 eslcudere 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
```

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
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

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
```

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
```

```
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
```

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)

# elemet-wise
my_mat*my_mat
```

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

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)
```

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**)

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

- Creazione
- Indicizzazione
- Esempi
- Combinare Dataframes
- Esportazione e importazione dati
- Ora però facciamo un po' di pratica!

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  1.9683616
2      2       b  0.1965249
3      3       c  0.2451950
4      4       d -0.9764253
```

Attributi

Il **dataframe** ha sia gli attributi della lista ovvero i ***names*** ma anche gli attributi della matrice ovvero le ***dimensioni*** (righe e colonne)

```
attributes(my_df)
```

```
$names
[1] "numeri"  "lettere" "normale"

$class
[1] "data.frame"

$row.names
[1] 1 2 3 4
```

```
dim(my_df)
```

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  1.968 0.197 0.245 -0.976
```

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

- Creazione
- Indicizzazione
- Esempi
- Combinare Dataframes
- Esportazione e importazione dati
- Ora però facciamo un po' di pratica!

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
```

```
[1] 1
```

Indicizzazione $

```
my_df$numeri # estraggo la prima colonna del data.frame
```

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

```
my_df$numeri[1] #  estraggo il primo elemento della prima colonna
```

```
[1] 1
```

Indicizzazione - Operatori relazionali

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

```
my_df
```

```
  numeri lettere    normale
1      1       a  1.9683616
2      2       b  0.1965249
3      3       c  0.2451950
4      4       d -0.9764253
```

```
# includo solo le righe in cui alla colonna1 i valori sono maggi
my_df[my_df$numeri > 2, ]
```

```
  numeri lettere    normale
```

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

- Creazione
- Indicizzazione
- Esempi
- Combinare Dataframes
- Esportazione e importazione dati
- Ora però facciamo un po' di pratica!

Esempi

```
my_df
```

```
  numeri lettere    normale
1      1       a  1.9683616
2      2       b  0.1965249
3      3       c  0.2451950
4      4       d -0.9764253
```

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

```
  numeri lettere  normale
3      3       c 0.245195
```

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

```
character(0)
```

Indicizzazione subset()

```
str(iris)
```

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ..
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ..
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1
```

```
subset(iris, subset = Species == "setosa" & Petal.Length > 1.7)
```

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
25          4.8         3.4          1.9         0.2  setosa
45          5.1         3.8          1.9         0.4  setosa
```

Equivalente a:

```
iris[iris$Species == "setosa" & iris$Petal.Length > 1.7,]
```

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
25          4.8         3.4          1.9         0.2  setosa
45          5.1         3.8          1.9         0.4  setosa
```

subset(df, select = ...)

E' possibile anche selezionare colonne piuttosto che righe attraverso l'argomento ***select***:

```
subset(iris, select = c(Sepal.Length, Species))
```

```
#visualizzo le prime due righe attraverso il comando head
head(subset(iris, select = c(Sepal.Length, Species)), n = 3)
```

```
  Sepal.Length Species
1          5.1  setosa
2          4.9  setosa
3          4.7  setosa
```

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

Possiamo anche combinare le due cose:

```
subset(iris, subset = Species == "setosa" & Sepal.Length > 4, sel
```

```
head(subset(iris, subset = Species == "setosa" & Sepal.Length > 4
```

```
  Sepal.Length Species
1          5.1  setosa
2          4.9  setosa
3          4.7  setosa
```

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 variablie che è la somma di Length e Width
iris$somma = iris$Sepal.Length + iris$Sepal.Width

str(iris)
```

```
'data.frame':   150 obs. of  6 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ..
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ..
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1
 $ somma       : num  8.6 7.9 7.9 7.7 8.6 9.3 8 8.4 7.3 8 ...
```

Si applicano gli stessi concetti che abbiamo visto per i vettori, potete quindi sia creare che modificare variabili.

```
my_df = data.frame(num = 1:4, let = letters[1:4])
my_df
```

```
  num let
1   1   a
2   2   b
3   3   c
4   4   d
```

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

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

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

- Creazione
- Indicizzazione
- Esempi
- Combinare Dataframes
- Esportazione e importazione dati
- Ora però facciamo un po' di pratica!

Combinare Dataframes

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

```
# primo dataframe
str(my_df)
```

```
'data.frame':   4 obs. of  3 variables:
 $ num : num  2 3 4 5
 $ let : chr  "a" "b" "c" "d"
 $ both: chr  "2_a" "3_b" "4_c" "5_d"
```

```
# creo un secondo dataframe
my_df2 = data.frame(num = 4:7, lett = letters[1:4],
                    both = paste(4:7,letters[1:4], sep = "_"))

str(my_df2)
```

Unisco i due dataframes

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

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

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

```
str(my_df)
```

```
'data.frame':   4 obs. of  3 variables:
 $ num : num  2 3 4 5
 $ let : chr  "a" "b" "c" "d"
 $ both: chr  "2_a" "3_b" "4_c" "5_d"
```

```
str(my_df2)
```

Sistemo i nomi

```
names(my_df2)
```

```
[1] "num"  "lett" "both"
```

```
names(my_df)
```

```
[1] "num"  "let"  "both"
```

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

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

```
'data.frame':   8 obs. of  3 variables:
 $ num : num  2 3 4 5 4 5 6 7
```

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

```
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
                             rlnorm(n = 400, meanlog = -.7, sdlog
```

Dataframe contente l'età:

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

```
str(df_rt) # struttura dataframe tempi di reazione
```

```
'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  0.212 0.306 0.244 0.259 0.343 ...
```

```
head(df_rt)
```

```
  subj cond        rt
1 caio easy 0.2124034
2 caio easy 0.3061848
3 caio easy 0.2437760
4 caio easy 0.2587453
5 caio easy 0.3433939
6 caio easy 0.3356226
```

```
str(df_age) # struttura dataframe età
```

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  0.212 0.306 0.244 0.259 0.343 ...
 $ age : num  20 20 20 20 20 20 20 20 20 20 ...
```

la funzione `_join()`:

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

df_all_2 = left_join(x = df_rt, y = df_age, by = c("subj")) # es
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  0.212 0.306 0.244 0.259 0.343 ...
 $ age : num  20 20 20 20 20 20 20 20 20 20 ...
```

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

- Creazione
- Indicizzazione
- Esempi
- Combinare Dataframes
- Esportazione e importazione dati
- Ora però facciamo un po' di pratica!

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)
write_xlsx(df_age, path = "data/df_age.xlsx")
save(df_age, file = "data/df_age.rda") # formato R
```

Importo

```r
library(readxl) # carico il pacchetto readxl

df_rt_impo = read_csv("data/df_rt.csv") #utilizza il pacchetto r
df_age_impo = read_xlsx("dat/df_age.xlsx")
load(file ="data/df_age.rda") # formato R
```

Controllo la struttura ed il tipo di dati

```
str(df_rt_impo)
```

```
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] 0.212 0.306 0.244 0.259 0.343 ...
 - attr(*, "spec")=
  .. cols(
  ..   subj = col_character(),
  ..   cond = col_character(),
  ..   rt = col_double()
  .. )
 - attr(*, "problems")=<externalptr>
```

```
df_rt_impo$subj = as.factor(df_rt_impo$subj)
```

1 Vettori

2 Fattori

3 Matrici

4 Dataframe

- Creazione
- Indicizzazione
- Esempi
- Combinare Dataframes
- Esportazione e importazione dati
- Ora però facciamo un po' di pratica!

Ora però facciamo un po' di pratica!

Aprite e tenete aperto questo link:

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