Testing Psicologico

Lezione 1B - Matrici e Dataframe

Filippo Gambarota

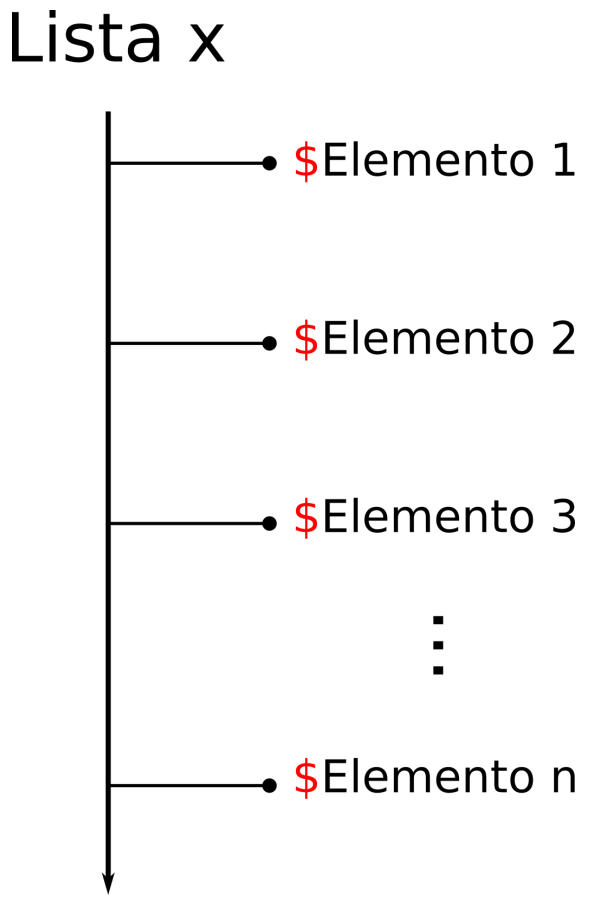
@Università di Padova

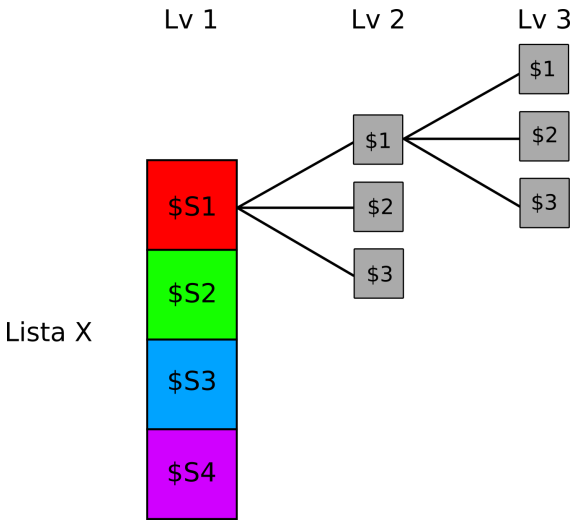
2022-2023

Liste

Piccolo ripasso...

In R la lista è la struttura dati più versatile (meno strutturata 😄) e utile.





Piccolo ripasso...

```
list(elemento1, elemento2, elemento3) # lista normale
list(nome1 = elemento2, nome2 = elemento2, nome3 = elemento3) # lista named
```

```
el1 <- runif(100)
el2 <- rep(letters[1:10], 3)
el3 <- iris
my_list <- list(vec1 = el1, vec2 = el2, data = el3)

names(my_list)
```

```
## [1] "vec1" "vec2" "data"
```

```
length(my_list)
```

```
## [1] 3
```

```
str(my_list)
```

```
## List of 3
##  $ vec1: num [1:100] 0.0969 0.5048 0.466 0.1809 0.4052 ...
##  $ vec2: chr [1:30] "a" "b" "c" "d" ...
##  $ data:'data.frame':    150 obs. of  5 variables:
##   ..$ Sepal.Length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##   ..$ Sepal.Width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##   ..$ Petal.Length: num [1:150] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##   ..$ Petal.Width : num [1:150] 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 1 1 1 1 1 1 1 1 ...
```

Liste

Accediamo/modifichiamo gli elementi della lista:

```
my_list$vec1 # con il dollaro + nome
```

```
##  [1] 0.09694092 0.50479379 0.46597238 0.18094735 0.40516422 0.54075926 0.31539841
##  [8] 0.23473735 0.31794195 0.87176612
##  [ reached getOption("max.print") -- omitted 90 entries ]
```

```
my_list[1] # con la parentesi quadra
```

```
## $vec1
##  [1] 0.09694092 0.50479379 0.46597238 0.18094735 0.40516422 0.54075926 0.31539841
##  [8] 0.23473735 0.31794195 0.87176612
##  [ reached getOption("max.print") -- omitted 90 entries ]
```

```
my_list[[1]] # con la doppia parentesi quadra
```

```
##  [1] 0.09694092 0.50479379 0.46597238 0.18094735 0.40516422 0.54075926 0.31539841
##  [8] 0.23473735 0.31794195 0.87176612
##  [ reached getOption("max.print") -- omitted 90 entries ]
```

```
my_list[[1]] <- nuovoelemento # sovrascrivo il primo elemento
my_list[[4]] <- nuovoelemento # aggiungo un elemento
my_list[[length(my_list) +  1]] <- nuovoelemento # più raffinato
my_list <- append(my_list, list(nuovoelemento)) # usando la funzione append
my_list <- c(my_list, list(nome = nuovoelemento)) # usando la funzione c
```

Esercizi

1. Create una lista *named* che contenga
   - una sequenza di 20 numeri partendo da 3 e incrementando di 1.33 [elemento chiamato `el1`]
   - le lettere dell'alfabeto (vedi `letters`) ripetute tutte 2 volte [elemento chiamato `el2`]
   - 100 numeri casuali tra 1 e 100 (vedi il comando `runif`) [elemento chiamato `el3`]
2. Accedete al secondo elemento della lista
3. Aggiungete un quarto elemento con 10 numeri casuali da 1 a 100 (vedi `runif`)
4. Sostituite il terzo elemento con un'altra lista (lista nested) formata da 2 elementi:
   - un vettore numerico con i numeri da 1 a 30
   - le prime 10 lettere dell'alfabeto

Soluzioni

```r
# 1
my_list <- list(
  sequenza = seq(3, by = 1.33, length.out = 10),
  lettere = rep(letters, 2),
  iris = iris,
  normale01 = rnorm(100, mean = 0, sd = 1)
)

# 2
my_list[[2]]
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
##  [ reached getOption("max.print") -- omitted 42 entries ]
```

```r
# 2
my_list$lettere # se conosco il nome
```

```
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
##  [ reached getOption("max.print") -- omitted 42 entries ]
```

```r
# 3
my_list <- c(my_list, list(new_normale = rnorm(10, 10, 0)))
```

Soluzioni

```
# 4
my_list[[3]] <- list(1:30, letters[1:10])
my_list[[3]]
```

```
## [[1]]
##  [1]  1  2  3  4  5  6  7  8  9 10
##  [ reached getOption("max.print") -- omitted 20 entries ]
##
## [[2]]
##  [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

```
str(my_list)
```

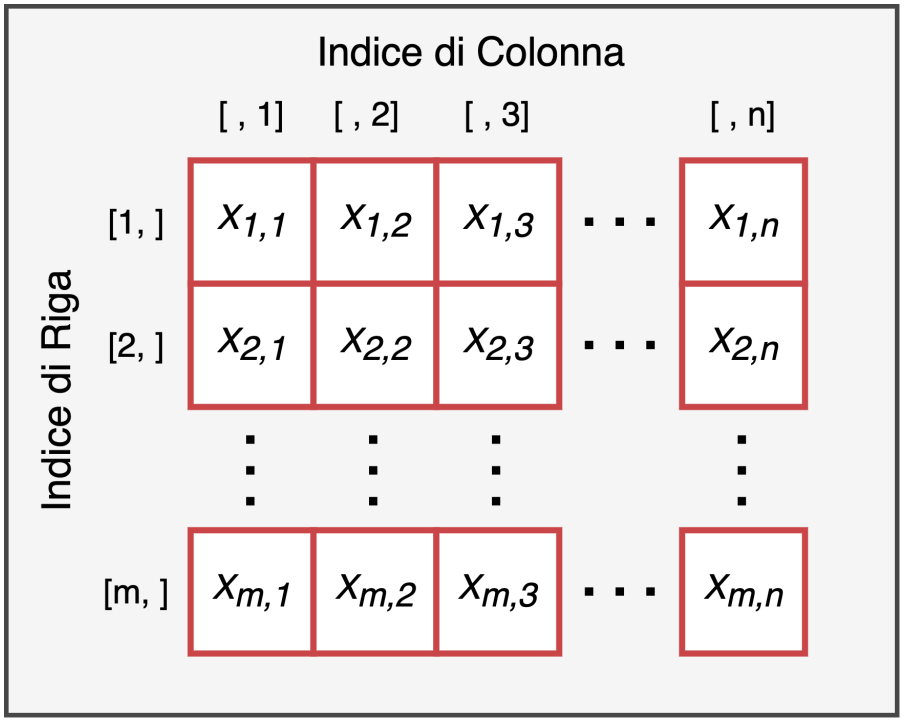
```
## List of 5
##  $ sequenza   : num [1:10] 3 4.33 5.66 6.99 8.32 ...
##  $ lettere    : chr [1:52] "a" "b" "c" "d" ...
##  $ iris       :List of 2
##   ..$ : int [1:30] 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ : chr [1:10] "a" "b" "c" "d" ...
##  $ normale01  : num [1:100] -0.838 -0.104 -0.313 -0.908 1.735 ...
##  $ new_normale: num [1:10] 10 10 10 10 10 10 10 10 10 10
```

Matrici

Dubbi/Domande? 🤔

Piccolo ripasso...

- Le matrici sono una struttura dati **bidimensionale** che contengono **una sola tipologia** di elementi
- Le proprietà fondamentali sono la tipologia (`str(matrice)`) e le dimensioni (`dim(matrice)`, `ncol(matrice)`, `nrow(matrice)`)



Indicizzazione Matrici

Indicizzazione Matrici

- Le matrici seguono la stessa logica dei vettori in termini di indicizzazione con la differenza di ragionare in modo bidimensionale $[righe, colonne]$
- Possiamo però anche usare l'indicizzazione logica e ovviamente intera

```
mat <- matrix(data = sample(1:30, 25, replace = TRUE),
              nrow = 5,
              ncol = 5,
              byrow = TRUE)
mat > 10 # tutti i numeri > 10
```

```
##       [,1]  [,2]  [,3] [,4]  [,5]
## [1,] FALSE  TRUE  TRUE TRUE FALSE
## [2,] FALSE FALSE  TRUE TRUE  TRUE
##  [ reached getOption("max.print") -- omitted 3 rows ]
```

```
mat == 10 # tutti i numeri uguali a 10
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,]  TRUE FALSE FALSE FALSE FALSE
## [2,] FALSE FALSE FALSE FALSE FALSE
##  [ reached getOption("max.print") -- omitted 3 rows ]
```

Indicizzazione Matrici

Possiamo quindi usare questa matrice logica (come con il vettore logico) per selezionare gli elementi:

```
mat[mat > 10]
```

```
##  [1] 19 24 24 15 25 18 12 22 11 21
##  [ reached getOption("max.print") -- omitted 7 entries ]
```

```
mat[mat == 10]
```

```
## [1] 10 10
```

Indicizzazione Matrici

Come vedete il risultato è un *vettore*. In pratica è come *srotolare* la matrice e poi trattarla esattamente come un vettore:

```
matv <- c(mat) # la funzione c permette di srotolare la matrice, anche con as.vector(mat)
matv
```

```
##  [1] 10  6 19  1  9 24  8 24 15 25
##  [ reached getOption("max.print") -- omitted 15 entries ]
```

```
matv[matv > 10] # equivalente a mat[mat > 10]
```

```
##  [1] 19 24 24 15 25 18 12 22 11 21
##  [ reached getOption("max.print") -- omitted 7 entries ]
```

Matrici - Esercizi

1.Create la seguente **matrice**:

$$\begin{bmatrix} 3 & 5 & 11 \\ 2 & 99 & 4 \\ 2 & 55 & 100 \\ 1 & 0 & 3 \end{bmatrix}$$

2.Data la matrice 1:

- accedere al numero di dimensioni
- accedere alla terza colonna
- accedere agli elementi $x_1 = [3, 1]$ e $x_2 = [4, 2]$, cosa notate?
- estraete dalla matrice tutti i numeri maggiori di 50
- estraete dalla matrice tutti i numeri pari (vedi l'operatore `%%`)
- sostituite tutti gli elementi dispari con il numero 0
- aggiungete una colonna con i numeri $[1, 2, 3, 4]$
- togliete la colonna 2

3.Creare la seguente **matrice**, cosa notate?

$$\begin{bmatrix} a & b & 11 \\ 1 & 22 & 4 \\ 4 & 55 & h \\ 1 & d & 3 \end{bmatrix}$$

4.Create una matrice formata da 30 lettere dell'alfabeto in modo random (vedi il comando `sample()` e l'oggetto `letters`) con numero di righe e colonne a vostra scelta. cosa notate rispetto alla matrice 1?

Soluzioni

```r
# 1
mat <- matrix(data = c(3,5,11,2,99,4,2,55,100,1,0,3),
              nrow = 4,
              ncol = 3,
              byrow = TRUE)

# 2
dim(mat)
```

```
## [1] 4 3
```

```r
mat[, 3] # mat[1:4, ], cosa cambia?
```

```
## [1]  11   4 100   3
```

```r
mat[3, 1]
```

```
## [1] 2
```

```r
mat[5, 2]
```

```
## Error in mat[5, 2]: subscript out of bounds
```

Soluzioni

```
# ..2
mat[mat > 50]
```

```
## [1]  99  55 100
```

```
mat[mat %% 2 == 0]
```

```
## [1]   2   2   0   4 100
```

```
mat[mat %% 2 != 0] <- 0
mat <- cbind(mat, c(1, 2, 3, 4))
mat[, -2]
```

```
##      [,1] [,2] [,3]
## [1,]    0    0    1
## [2,]    2    4    2
## [3,]    2  100    3
##  [ reached getOption("max.print") -- omitted 1 row ]
```

```
# 3
mat2 <- matrix(c("a", "b", 11, 1, 22, 4, 4, 55, "h", 1, "d", 3),
               nrow = 4,
               byrow = TRUE)
mat2
```

```
##      [,1] [,2] [,3]
## [1,] "a"  "b"  "11"
## [2,] "1"  "22" "4"
## [3,] "4"  "55" "h"
##  [ reached getOption("max.print") -- omitted 1 row ]
```

Soluzioni

```r
# 4
mat3 <- matrix(sample(letters, 30, replace = TRUE),
               nrow = 6,
               ncol = 5)
mat3
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,] "n"  "z"  "k"  "p"  "q"
## [2,] "q"  "q"  "w"  "f"  "l"
##  [ reached getOption("max.print") -- omitted 4 rows ]
```

```r
mat3 + 2
```

```
## Error in mat3 + 2: non-numeric argument to binary operator
```

```r
mat3 * 2
```

```
## Error in mat3 * 2: non-numeric argument to binary operator
```

```r
mat3 == "a"
```

```
##       [,1]  [,2]  [,3]  [,4]  [,5]
## [1,] FALSE FALSE FALSE FALSE FALSE
## [2,] FALSE FALSE FALSE FALSE FALSE
##  [ reached getOption("max.print") -- omitted 4 rows ]
```

Dataframe

Piccolo ripasso...

- Il dataframe è una struttura dati **bidimensionale** (come la matrice), può contenere **più tipologie di dati** (come la lista)
- E' un tipo particolare di **lista** dove la **lunghezza** di ogni elemento è fissa (vincolo) portando ad una **struttura rettangolare**
- E' la *traduzione* in codice del foglio di calcolo Excel

Piccolo ripasso...

Ci sono diversi dataframe già presenti in R come oggetti. Vediamo quello più semplice ovvero `iris`:

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
##  [ reached 'max' / getOption("max.print") -- omitted 4 rows ]
```

```
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 1 1 1 1 1 1 1 1 ...
```

```
class(iris)
```

```
## [1] "data.frame"
```

Piccolo ripasso...

Per accedere al dataframe usiamo un mix tra funzioni per le matrici (da cui prende la struttura rettangolare) e liste (da cui prende la flessibilità del tipo di dato):

```
iris$Sepal.Length # prima colonna/elemento
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
##  [ reached getOption("max.print") -- omitted 140 entries ]
```

```
iris[[1]] # prima colonna/elemento
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
##  [ reached getOption("max.print") -- omitted 140 entries ]
```

```
iris[, 1] # prima colonna
```

```
##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
##  [ reached getOption("max.print") -- omitted 140 entries ]
```

Indicizzazione dataframe

Indicizzazione dataframe

E' importante capire che a prescindere dalla complessità della struttura dati (vettore vs dataframe) quando selezioniamo delle righe/colonne non facciamo altro che *combinare operazioni logiche*, ottenere un vettore di `TRUE`/`FALSE` o di `interi` e con questo vettore indicare quali righe/colonne selezionare.

```
my_sel_log <- iris$Species == "Setosa"
my_sel_log # vettore logico TRUE/FALSE
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [ reached getOption("max.print") -- omitted 140 entries ]
```

```
my_sel_int <- which(my_sel_log) # vettore di interi

iris[my_sel_log, ] # selezione logica
```

```
## [1] Sepal.Length Sepal.Width  Petal.Length Petal.Width  Species
## <0 rows> (or 0-length row.names)
```

```
iris[my_sel_int, ] # selezione intera
```

```
## [1] Sepal.Length Sepal.Width  Petal.Length Petal.Width  Species
## <0 rows> (or 0-length row.names)
```

Extra: Importare dati

Importare dati

- La maggior parte delle analisi dati prevede di importare partendo da formati diversi (`xlsx`, `csv`, `sav`, `txt`, etc.) un dataset.
- Importare i dati è tutt'altro che banale e richiede una comprensione di come i vari formati codificano le informazioni fondamentali, in particolare la delimitazione dei valori
- `csv` ad esempio significa **c**omma **d**elimited **v**alues dove i valori sono delimitati da una virgola. R deve sapere il tipo di file e il delimitatore per leggere correttamente i dati

Per approfondire questo documento è una buona introduzione

Esempio

Esempio

```r
# importiamo i dati
dat <- read.csv("../../data/pazienti.csv", sep = ",", header = TRUE, fileEncoding="UTF-8-BOM")

str(dat) # struttura
```

```
## 'data.frame':    30 obs. of  6 variables:
##  $ sogg      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ regione   : chr  "Veneto" "Piemonte" "Lombardia" "Piemonte" ...
##  $ cl.sociale: chr  "Bassa" "Media" "Media" "Bassa" ...
##  $ ansia     : num  5.1 5.5 3.8 4.5 5.4 0.7 4.6 6 4.5 5.8 ...
##  $ eta       : int  58 48 21 57 39 52 42 48 53 32 ...
##  $ disturbo  : chr  "schizofrenia" "nevrosi" "schizofrenia" "nevrosi" ...
```

```r
nrow(dat) # numero di righe (osservazioni)
```

```
## [1] 30
```

```r
ncol(dat) # numero di colonne (variabili)
```

```
## [1] 6
```

```r
colnames(dat) # nomi delle colonne (variabili)
```

```
## [1] "sogg"       "regione"    "cl.sociale" "ansia"      "eta"        "disturbo"
```

Esempio

Lavorare in un dataframe segue la stessa logica di un foglio excel. Possiamo **filtrare** le righe e/o colonne in funzione di determinate *condizioni*:

```
# seleziono solo i pazienti con nevrosi e tutte le colonne
dat[dat$disturbo == "nevrosi", ]
```

```
##   sogg  regione cl.sociale ansia eta disturbo
## 2    2 Piemonte      Media   5.5  48  nevrosi
##  [ reached 'max' / getOption("max.print") -- omitted 11 rows ]
```

```
# seleziono solo i pazienti con età maggiore di 30
dat[dat$eta > 30, ]
```

```
##   sogg regione cl.sociale ansia eta     disturbo
## 1    1  Veneto      Bassa   5.1  58 schizofrenia
##  [ reached 'max' / getOption("max.print") -- omitted 27 rows ]
```

```
# seleziono i pazienti con ansia maggiore di 3 E provenienti dal veneto
dat[dat$ansia > 3 & dat$regione == "Veneto", ]
```

```
##   sogg regione cl.sociale ansia eta     disturbo
## 1    1  Veneto      Bassa   5.1  58 schizofrenia
##  [ reached 'max' / getOption("max.print") -- omitted 2 rows ]
```

Esercizi

1. Importa il dataframe `pazienti_sc.csv` (attenzione al *separatore*)
2. Estrai la struttura, il numero di colonne/righe
3. Estrai le righe 1, 10, 15, e 30
4. Estrai le righe da 1 a 15 e la 1 e 4 colonna
5. Estrai le osservazioni di pazienti provenienti dalla Liguria O dal Piemonte di classe sociale Alta e disturbi NON fobici
6. Estrai le osservazioni con età compresa tra 20 e 45 anni

Soluzioni

```r
dat <- read.csv("../../data/pazienti_sc.csv", sep = ";", header = TRUE, fileEncoding="UTF-8-BOM") # import

# struttura, righe e colonne
str(dat)
```

```
## 'data.frame':    30 obs. of  6 variables:
##  $ sogg      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ regione   : chr  "Veneto" "Piemonte" "Lombardia" "Piemonte" ...
##  $ cl.sociale: chr  "Bassa" "Media" "Media" "Bassa" ...
##  $ ansia     : num  5.1 5.5 3.8 4.5 5.4 0.7 4.6 6 4.5 5.8 ...
##  $ eta       : int  58 48 21 57 39 52 42 48 53 32 ...
##  $ disturbo  : chr  "schizofrenia" "nevrosi" "schizofrenia" "nevrosi" ...
```

```r
nrow(dat)
```

```
## [1] 30
```

```r
ncol(dat)
```

```
## [1] 6
```

Soluzioni

```
dat[c(1, 10, 15, 30), ] # righe 1, 10, 15 e 30
```

```
##   sogg regione cl.sociale ansia eta     disturbo
## 1    1  Veneto      Bassa   5.1  58 schizofrenia
##  [ reached 'max' / getOption("max.print") -- omitted 3 rows ]
```

```
dat[1:15, c(1, 4)] # righe 1:15 e colonna 1 e 4
```

```
##   sogg ansia
## 1    1   5.1
## 2    2   5.5
## 3    3   3.8
## 4    4   4.5
## 5    5   5.4
##  [ reached 'max' / getOption("max.print") -- omitted 10 rows ]
```

```
dat[(dat$regione == "Liguria" | dat$regione == "Piemonte") & dat$cl.sociale == "Alta" & dat$disturbo != "fobico", ] # pazien
```

```
##   sogg regione cl.sociale ansia eta disturbo
## 7    7 Liguria       Alta   4.6  42  nevrosi
##  [ reached 'max' / getOption("max.print") -- omitted 4 rows ]
```

```
dat[dat$eta > 20 & dat$eta < 45, ] # eta compresa tra 20 e 45
```

```
##   sogg   regione cl.sociale ansia eta     disturbo
## 3    3 Lombardia      Media   3.8  21 schizofrenia
##  [ reached 'max' / getOption("max.print") -- omitted 11 rows ]
```

EDA - Exploratory Data Analysis

Nuove colonne

Possiamo aggiungere nuove informazioni (colonne) per aggiungere informazioni o modificare quelle esistenti:

```
# aggingiamo una colonna che indica alta o bassa ansia basandoci su un cut-off di 4
dat$ansia_cut <- ifelse(dat$ansia > 4, yes = "alta", no = "bassa")

# convertiamo la classe sociale in un fattore ordinato (scala ordinale)
dat$cl.sociale <- factor(dat$cl.sociale, ordered = TRUE)

str(dat)
```

```
## 'data.frame':    30 obs. of  7 variables:
##  $ sogg      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ regione   : chr  "Veneto" "Piemonte" "Lombardia" "Piemonte" ...
##  $ cl.sociale: Ord.factor w/ 3 levels "Alta"<"Bassa"<..: 2 3 3 2 2 3 1 2 3 2 ...
##  $ ansia     : num  5.1 5.5 3.8 4.5 5.4 0.7 4.6 6 4.5 5.8 ...
##  $ eta       : int  58 48 21 57 39 52 42 48 53 32 ...
##  $ disturbo  : chr  "schizofrenia" "nevrosi" "schizofrenia" "nevrosi" ...
##  $ ansia_cut : chr  "alta" "alta" "bassa" "alta" ...
```

Esplorazione

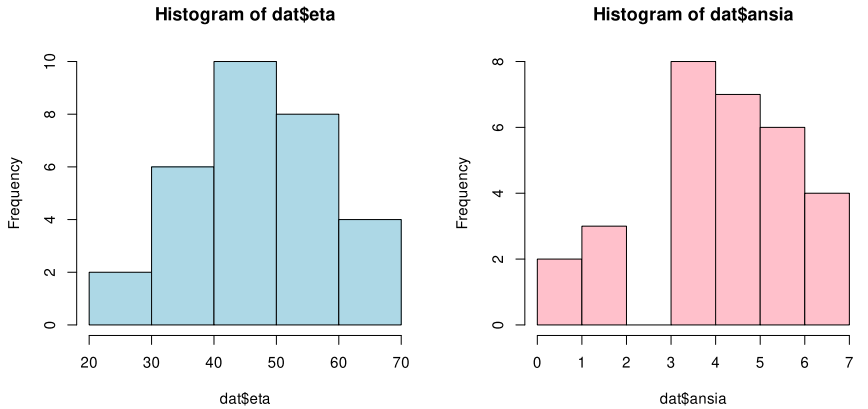
Ogni tipo di variabile è associata a determinate statistiche descrittive (e.g., *media* vs *frequenza*) e rappresentazioni grafiche (e.g., *barplot* vs *boxplot*).

- ha senso calcolare la media della variabile `disturbo`?
- ha senso calcolare le frequenze della variabile `ansia`?

Esplorazione

Facciamo un istogramma per le variabili numeriche:

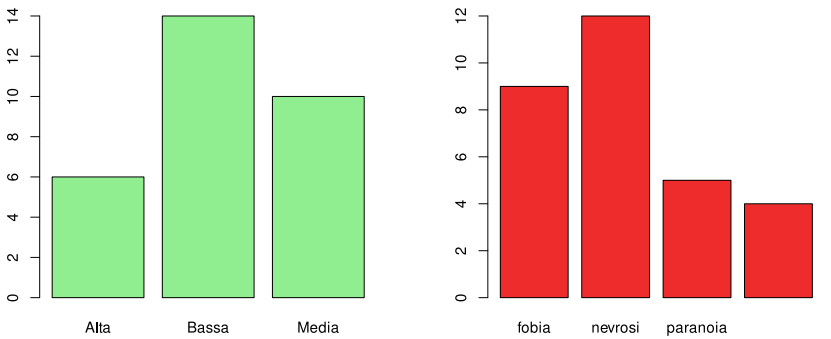
```
par(mfrow = c(1,2))
hist(dat$eta, col = "lightblue")
hist(dat$ansia, col = "pink")
```



Esplorazione

Facciamo un grafico a barre per le variabili categoriali/ordinali

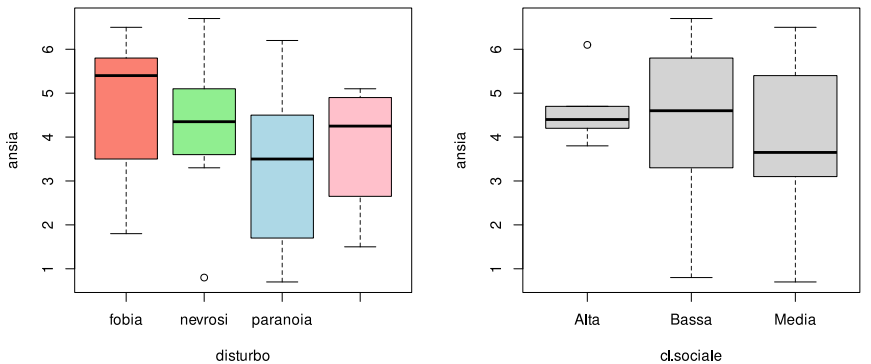
```
par(mfrow = c(1,2))
barplot(table(dat$cl.sociale), col = "lightgreen")
barplot(table(dat$disturbo), col = "firebrick2")
```



Explorazione - Grafici bi-variati

Possiamo vedere la distribuzione di una variabile numerica *in funzione* di una categoriale:

```
par(mfrow = c(1,2))
boxplot(ansia ~ disturbo, data = dat, col = c("salmon", "lightgreen", "lightblue", "pink"))
boxplot(ansia ~ cl.sociale, data = dat)
```



Esplorazione - Grafici bi-variati

Possiamo anche vedere la distribuzione di due variabili categoriali facendo un barplot ed una tabella di contingenza:

```
barplot(table(dat$cl.sociale, dat$disturbo), col = c("salmon", "lightgreen", "lightblue"))
legend("topright", legend=unique(dat$cl.sociale), pch=16, col = c("salmon", "lightgreen", "lightblue"))
```