

# Grafici, Funzioni, cenni di statistica

# Funzioni

- Funzioni come  $\min(x)$   $\max(x)$   $\text{mean}(x)$  etc.. sono funzioni predefinite di Matlab.
- Alcune funzioni di Matlab sono editabili nonostante siano predefinite.
- Funzione da un input, calcola un output
- Script Riutilizzabile
- A volte sottoprogramma
- Elementi costruttivi per grandi programmi
- L'ultima volta abbiamo visto la funzione  $\text{deg2rad}$

# Funzione Deg2rad

Si usa così :

Z = deg2rad (180)

Ans=3.1416

oppure

deg2rad (30)

ans=0.5236

```
function y = deg2rad(x)
% DEG2RAD Convert angles from degrees to radians
%
y = x * (pi/180);
```

## Regole per le Functions:

**function**  
(identificativo per un file  
funzione)

**[output]**  
Lista dei risultati  
della funzione

**Nome**  
Nome della funzione  
E del file (nome.m  
)

**Input**  
Lista delle variabili  
In ingresso

**Function [output] = nome(input)**

- **Input**

Serve per trasferire dati alla funzione dal workspace

Le variabili del Workspace non sono disponibili all'interno della funzione

Nel caso volessimo usare più input : bisogna separarli con delle virgole, l'ordine è preso in considerazione e quindi va rispettato

- **Output**

Utilizzato per restituire al Workspace i risultati della funzione

**Output multipli** : Separati da virgole dentro quadre, L'ordine è importante

Es [Value, Location]=max(x)

# options

- La variabile **nargin** conserva il numero dei parametri input della funzione
- La variabile **nargout** conserva il numero dei parametri output della funzione
- Per scrivere una funzione con un numero variabile di parametri input/output si devono usare come ultimo argomento le variabili **varargin/varargout**

# Esempio nargin

```
function [x0, y0] = myplot(x, y, npts, angle, subdiv)
% MYPLOT Plot a function.
% MYPLOT(x, y, npts, angle, subdiv)
% The first two input arguments are
% required; the other three have default values.
...
if nargin < 5, subdiv = 20; end
if nargin < 4, angle = 10; end
if nargin < 3, npts = 25; end
...
if nargin == 0
% plot(x, y)
else
x0 = x;
y0 = y;
end
```

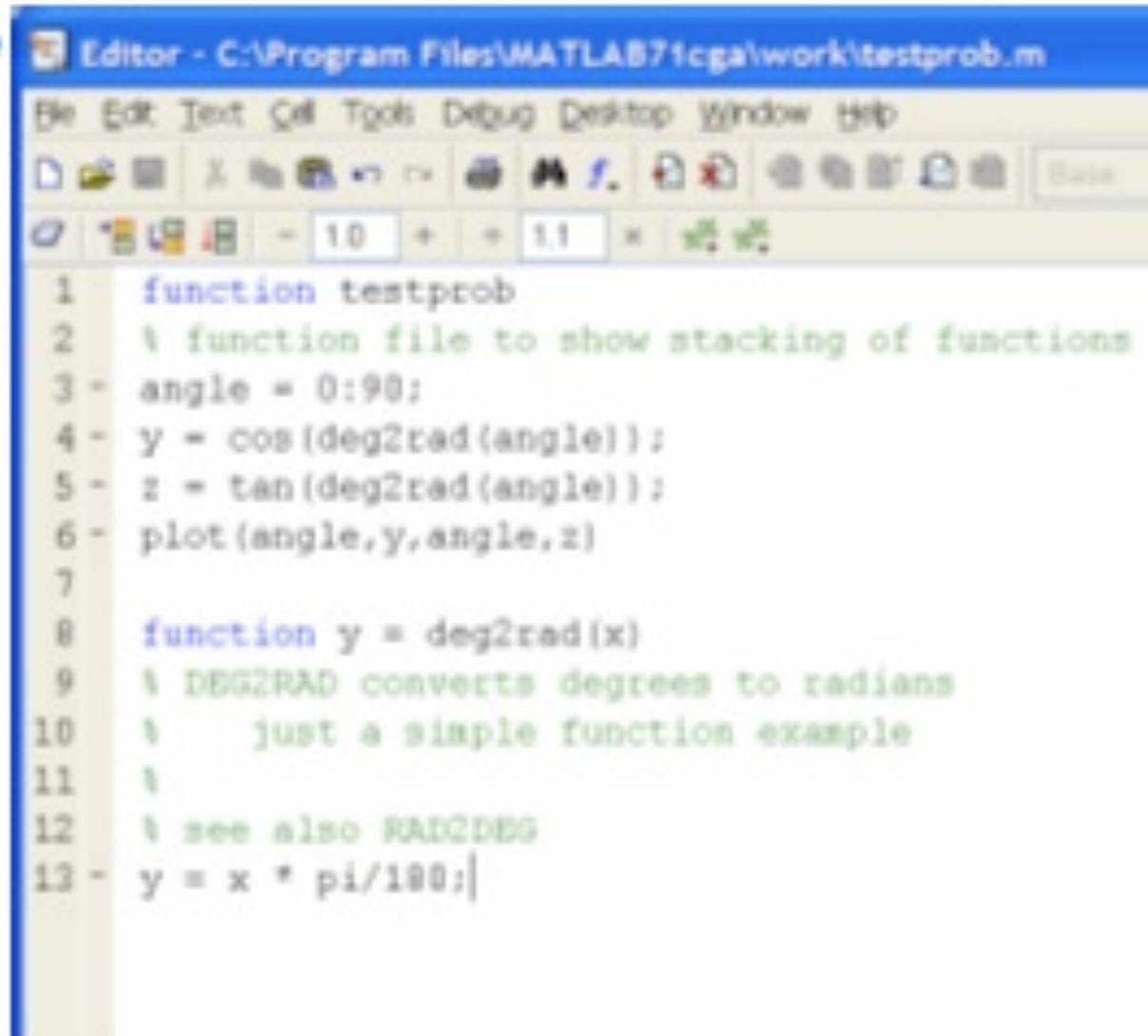
Se nargin == 1 significa che se la variabile di input è solo 1 allora dim viene calcolato utilizzando la formula  $dim = \min(\text{find}(\text{size}(x) \sim 1))$ ;

# Esempio nargout

```
function [x0, y0] = myplot(x, y, npts, angle, subdiv)
% MYPLOT Plot a function.
% MYPLOT(x, y, npts, angle, subdiv)
% The first two input arguments are
% required; the other three have default values.
...
if nargin < 5, subdiv = 20; end
if nargin < 4, angle = 10; end
if nargin < 3, npts = 25; end
...
if nargout == 0
% plot(x, y)
else
    x0 = x;
    y0 = y;
end
```

Se non viene specificato  
l'output allora di default  
**nargout** esegue  
l'istruzione plot(x,y)

- Una funzione può usare funzioni che vengono definite nello stesso file, di seguito.
- Non sono visibili all'esterno!



```
Editor - C:\Program Files\MATLAB71cga\work\testprob.m
File Edit Text Cell Tools Debug Desktop Window Help
[Icons]
- 1.0 + + 1.1 * [Icons]
1 function testprob
2 \ function file to show stacking of functions
3 - angle = 0:90;
4 - y = cos(deg2rad(angle));
5 - z = tan(deg2rad(angle));
6 - plot(angle,y,angle,z)
7
8 function y = deg2rad(x)
9 \ DEG2RAD converts degrees to radians
10 \ just a simple function example
11 \
12 \ see also RAD2DEG
13 - y = x * pi/180;|
```

**N.B : L'unica funzione accessibile dall'esterno è la prima**

# 😊 Esercizi 😊

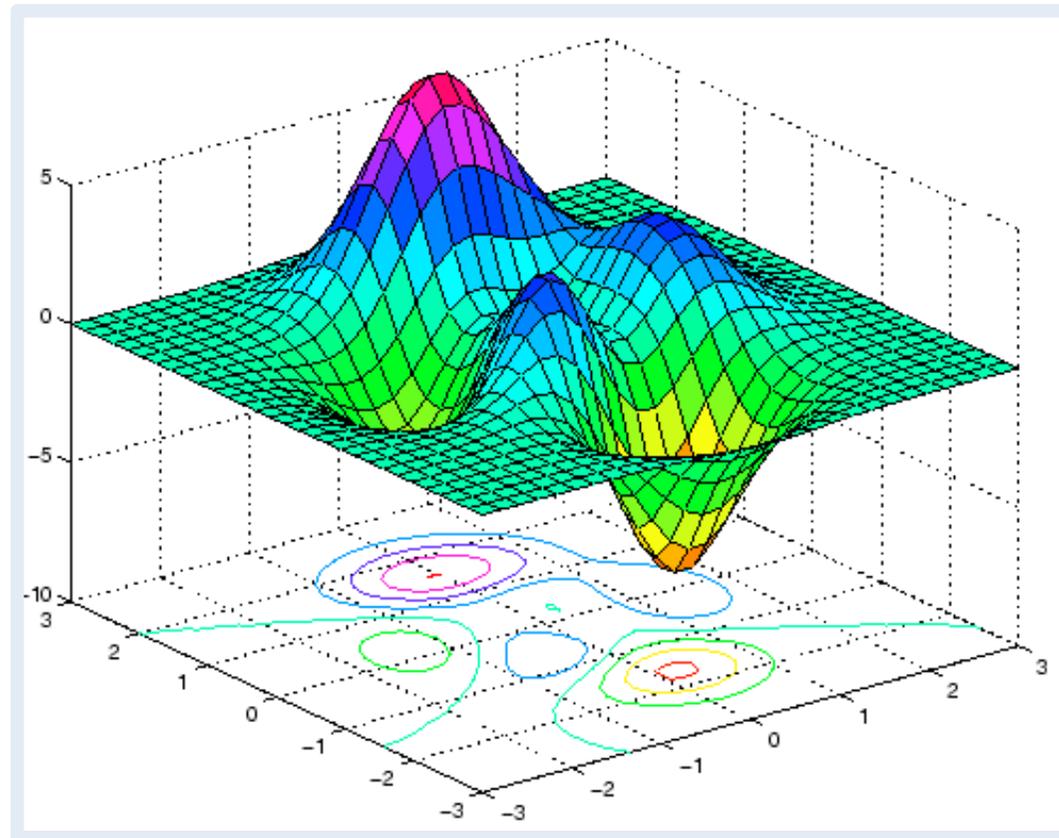
1. Data questa formula:  $V = \pi r^2 h$

Costruiamo una funzione che ci dia il volume dato il raggio e l'altezza del cilindro

1. Costruire una funzione che calcoli il perimetro e l'area di un triangolo dato base e altezza.

(Utilizzate wikipedia per le formule)

# Grafici con MatLab



# **Matlab: Creazione di grafici**

**Vedremo le 3 tipologie più semplici ed a noi utili:**

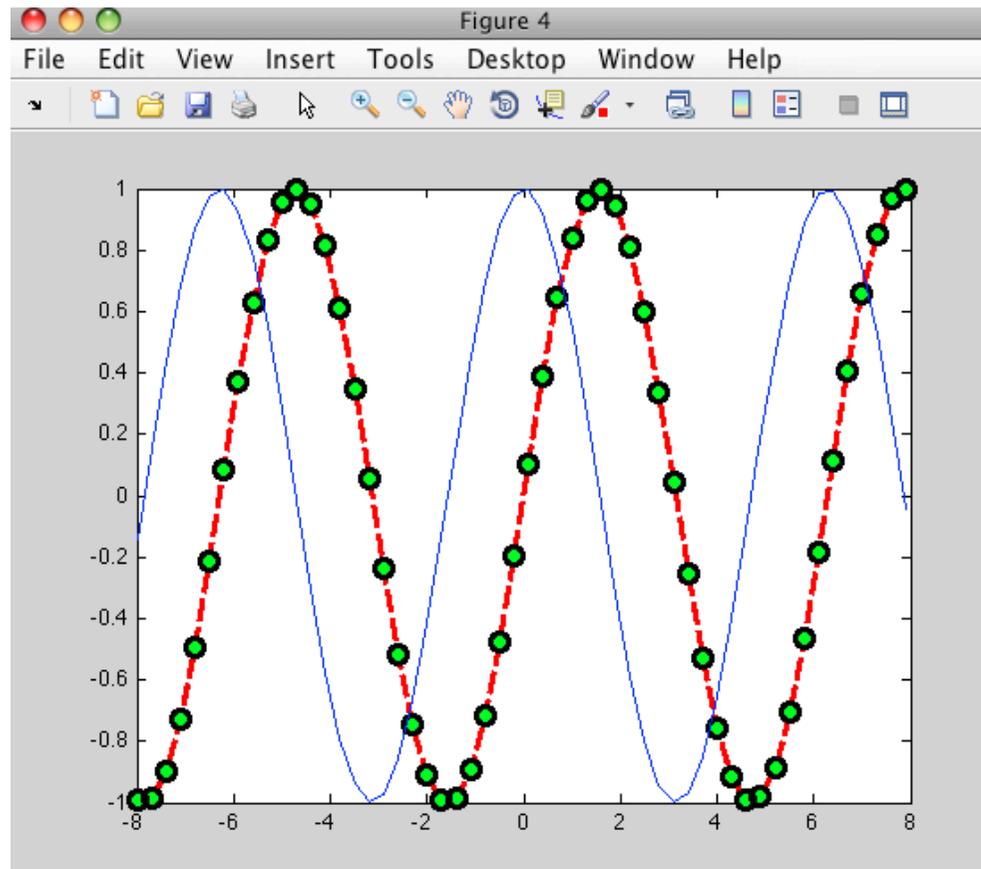
**- plot lineare**

**- scatter plot + regressione lineare**

**- istogramma a barre**

# Matlab: Creazione di grafici

“plot”: è la funzione base di visualizzazione dei dati 2D in MatLab. Creiamo un semplice grafico e vediamo come è possibile modificarlo.



# Matlab: Creazione di grafici

Per avere informazioni sulla “plot” possiamo usare il comando help. Un metodo alternativo e molte volte più esaustivo di comprendere le funzionalità di alcuni comandi è quello di usare il comando `doc(funzione)` che ci rimanda alla pagina web della documentazione di quel dato programma.

Dalla documentazione scopriamo che alcuni dei parametri che possono essere passati alla funzione plot sono:

```
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
plot(x,y, '--rs', 'LineWidth',2,...
      'MarkerEdgeColor','k',...
      'MarkerFaceColor','g',...
      'MarkerSize',10)
```

# doc plot

Help Navigator

Search for:  Go

Example: "plot tools" OR plot\* tools

Contents Index Search Results Demos

- Plots and Plotting Tools
  - Figures, Plots, and Graphs
  - Plotting Tools - Interactive Plotting
  - Example - Working with Plotting Tools
  - Example - Plotting from the Figure Palette
  - Example - Specifying a Data Source
  - Example - Generating M-Code to Reproduce a Graph
  - Editing Plots
  - Working in Plot Edit Mode
    - Figure Windows in Plot Edit Mode
    - Starting Plot Edit Mode
    - Exiting Plot Edit Mode
    - Selecting Objects in a Graph
    - Cutting, Copying, and Pasting Plot Objects
    - Moving and Resizing Objects
    - Setting Object Properties
    - Undo/Redo - Eliminating Mistakes
  - Saving Your Work
  - Data Exploration Tools
  - Annotating Graphs
  - Basic Plotting Commands
  - Creating Specialized Plots
  - Displaying Bit-Mapped Images
  - Printing and Exporting
  - Handle Graphics Objects
  - Figure Properties
  - Axes Properties
  - Examples
- 3-D Visualization
- Creating Graphical User Interfaces
- Function Reference

Title: plot :: Functions (MATLAB Function Reference)

imaginary component is ignored.

`plot(X1,Y1,...)` plots all lines defined by  $X_n$  versus  $Y_n$  pairs. If only  $X_n$  or  $Y_n$  is a matrix, the vector is plotted versus the rows or columns of the matrix, depending on whether the vector's row or column dimension matches the matrix. If  $X_n$  is a scalar and  $Y_n$  is a vector, disconnected line objects are created and plotted as discrete points vertically at  $X_n$ .

`plot(X1,Y1,LineStyle,...)` plots all lines defined by the  $X_n, Y_n, LineSpec$  triples, where [LineStyle](#) is a line specification that determines line type, marker symbol, and color of the plotted lines. You can mix  $X_n, Y_n, LineSpec$  triples with  $X_n, Y_n$  pairs:  
`plot(X1,Y1,X2,Y2,LineStyle,X3,Y3)`.

**Note** See [LineStyle](#) for a list of line style, marker, and color specifiers.

`plot(...,'PropertyName',PropertyValue,...)` sets properties to the specified property values for all [lineseries](#) graphics objects created by `plot`. (See the [Examples](#) section for examples.)

`plot(axes_handle,...)` plots into the axes with the handle `axes_handle` instead of into the current axes ([gca](#)).

`h = plot(...)` returns a column vector of handles to [lineseries](#) graphics objects, one handle per line.

**Backward-Compatible Version**

`hlines = plot('v6',...)` returns the handles to line objects instead of [lineseries](#) objects.

**Note** The `v6` option enables users of Version 7.x of MATLAB to create FIG-files that previous versions can open. It is obsolete and will be removed in a future version of MATLAB.

See [Plot Objects and Backward Compatibility](#) for more information.

# LineStyle

Line specification string syntax

## GUI Alternative

To modify the style, width, and color of lines on a graph, use the Property Editor, one of the plotting tools . For details, see [The Property Editor](#) in the MATLAB® Graphics documentation.

## Description

This page describes how to specify the properties of lines used for plotting. MATLAB graphics give you control over these visual characteristics:

- Line style
- Line width
- Color
- Marker type
- Marker size
- Marker face and edge coloring (for filled markers)

You indicate the line styles, marker types, and colors you want to display *specifiers*, detailed in the following tables:

## Line Style Specifiers

Specifier	Line Style
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line

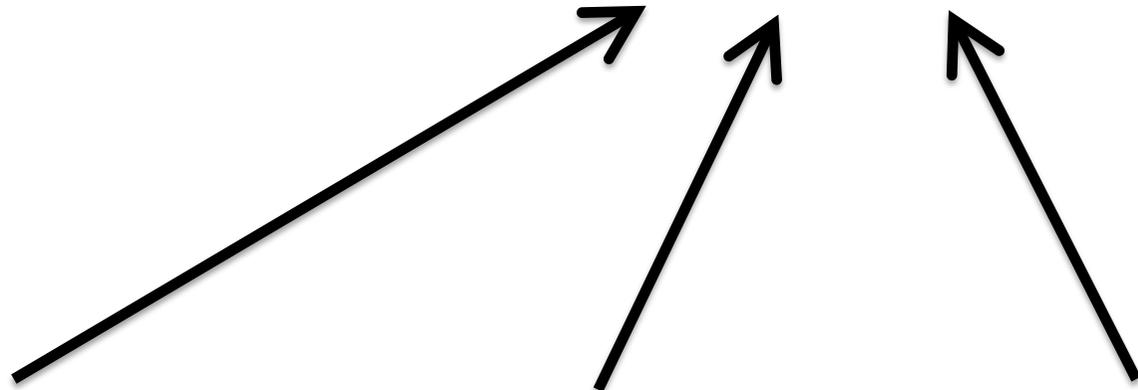
# Matlab: Creazione di grafici

Forma Generale : `plot(x,y,OBJ,OBJ,OBJ)`

Colore:  
K black  
R red  
B blue

Simbolo:  
. Point  
O Circle  
X x-mark  
s square  
d diamond

Tipo Linea:  
Solid  
: dotted  
-- dashed  
-. Dash dot

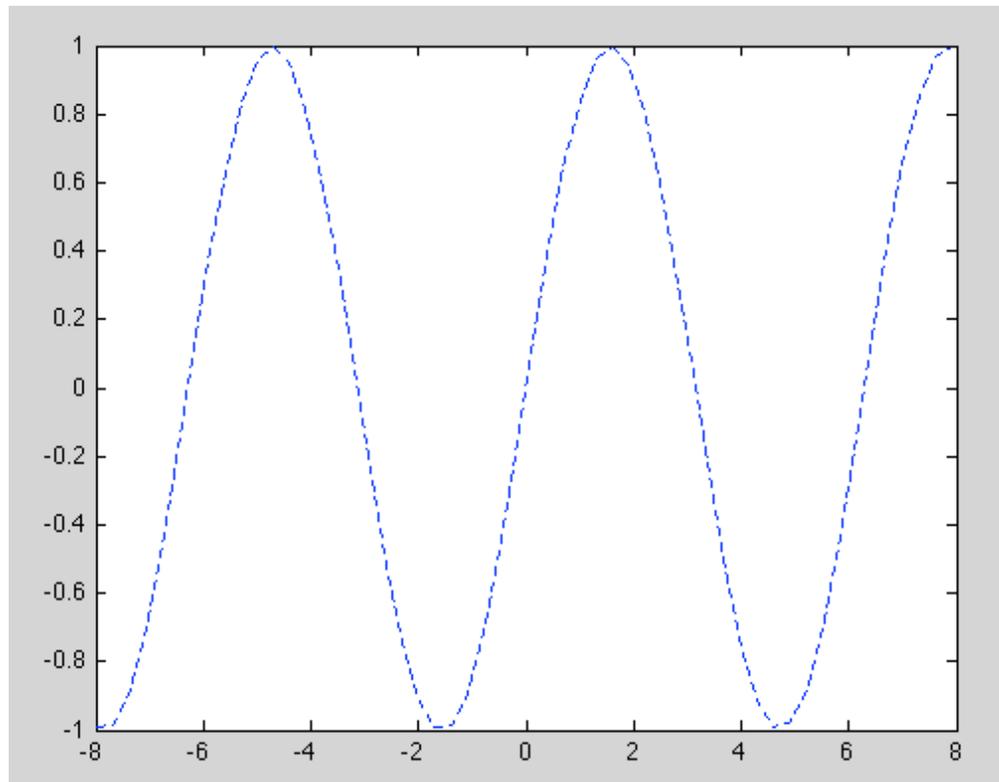


# Matlab: Creazione di grafici

```
figure  
plot ( x, y, '--' ) %tratteggiato
```

## Line Style Specifiers

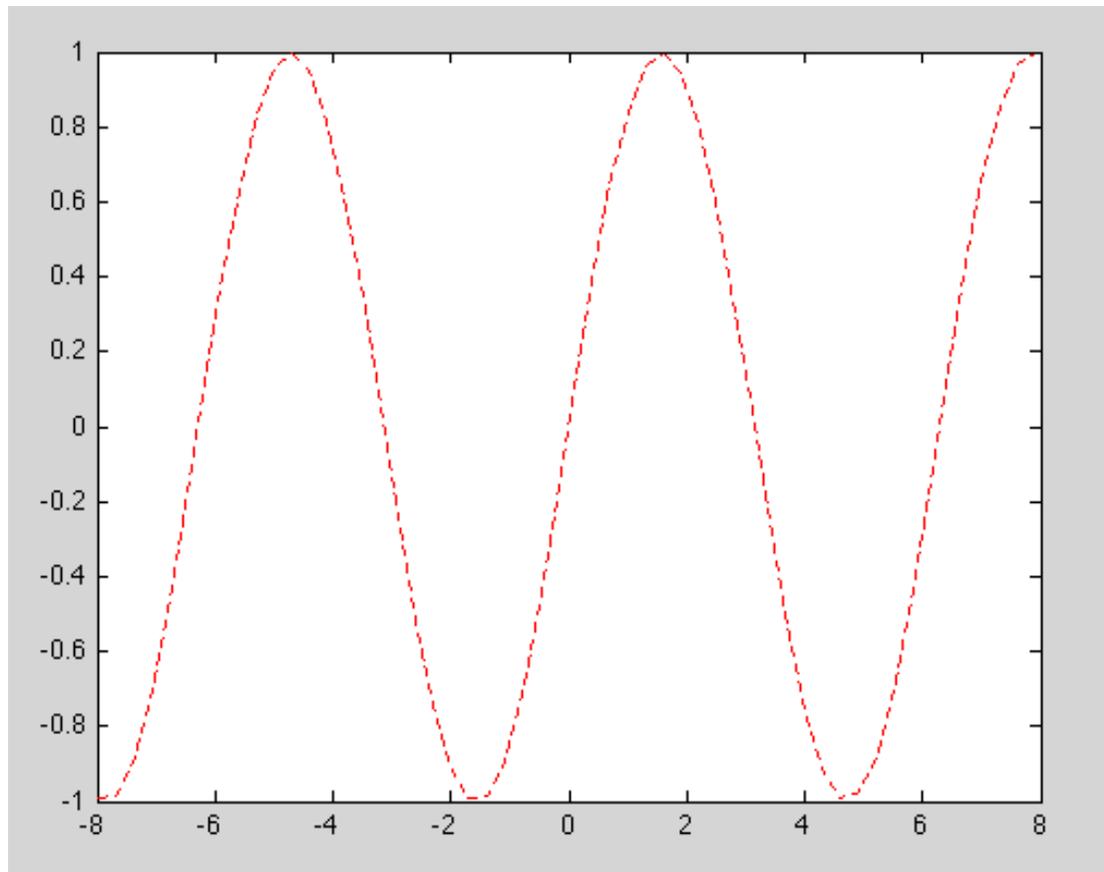
Specifier	Line Style
-	Solid line (default)
--	Dashed line
:	Dotted line
-.	Dash-dot line



# Matlab: Creazione di grafici

figure

```
plot ( x, y, '--r' ) %rosso
```



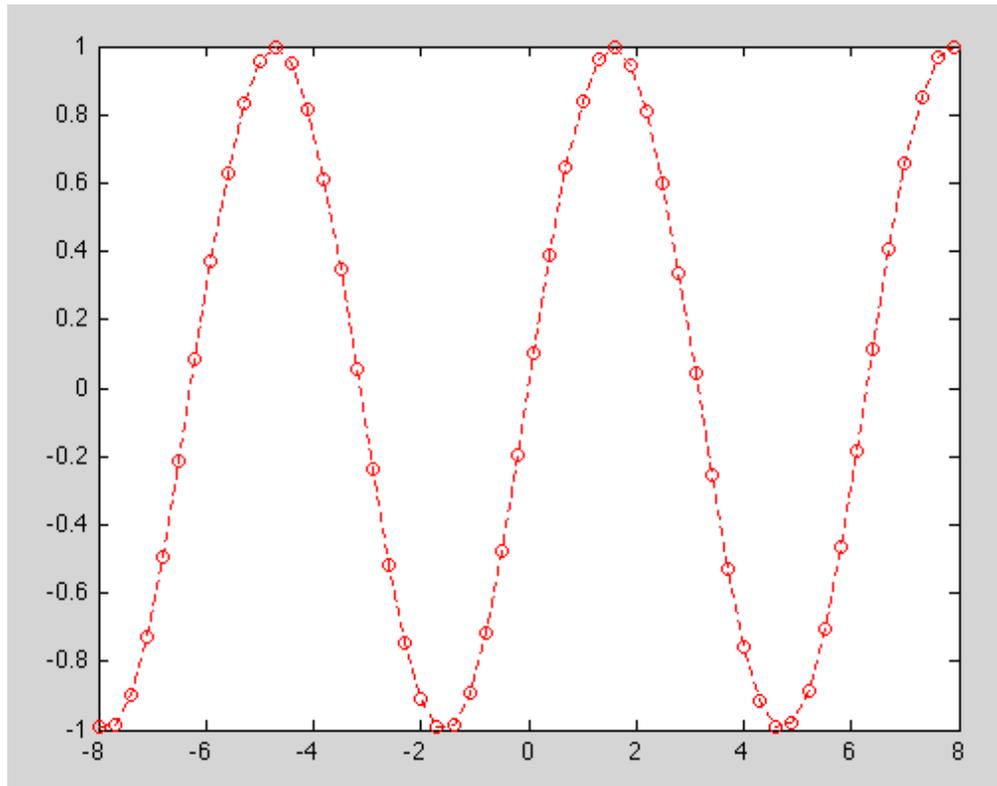
## Color Specifiers

Specifier	Color
r	Red
g	Green
b	Blue
c	Cyan
m	Magenta
y	Yellow
k	Black
w	White

# Matlab: Creazione di grafici

figure

plot ( x, y, '--or' ) %cerchi rossi



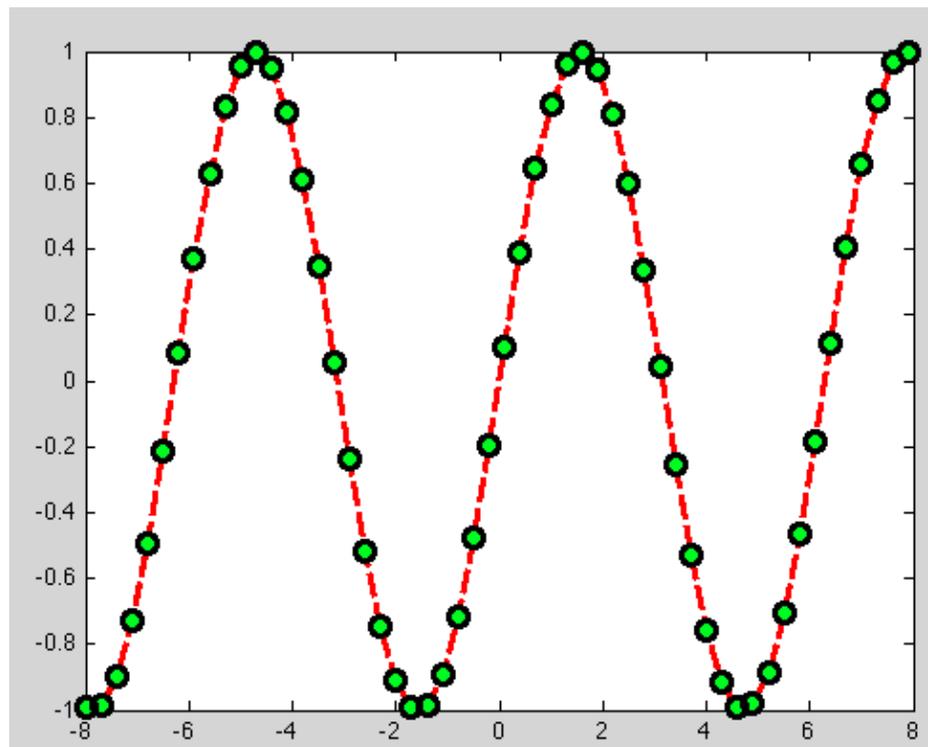
## Marker Specifiers

Specifier	Marker Type
+	Plus sign
o	Circle
*	Asterisk
.	Point
x	Cross
'square' or s	Square
'diamond' or d	Diamond
^	Upward-pointing triangle
v	Downward-pointing triangle
>	Right-pointing triangle
<	Left-pointing triangle
'pentagram' or p	Five-pointed star (pentagram)
'hexagram' or h	Six-pointed star (hexagram)

# Matlab: Creazione di grafici

Utilizzando le molte opzioni a disposizione possiamo modificare le caratteristiche estetiche del grafico a nostro piacimento.

```
plot(x,y,'--or','LineWidth',2,... %spessore della linea  
      'MarkerEdgeColor','k',... %colore bordo dei simboli  
      'MarkerFaceColor','g',... %colore interno dei simboli  
      'MarkerSize',8)           %grandezza simboli
```

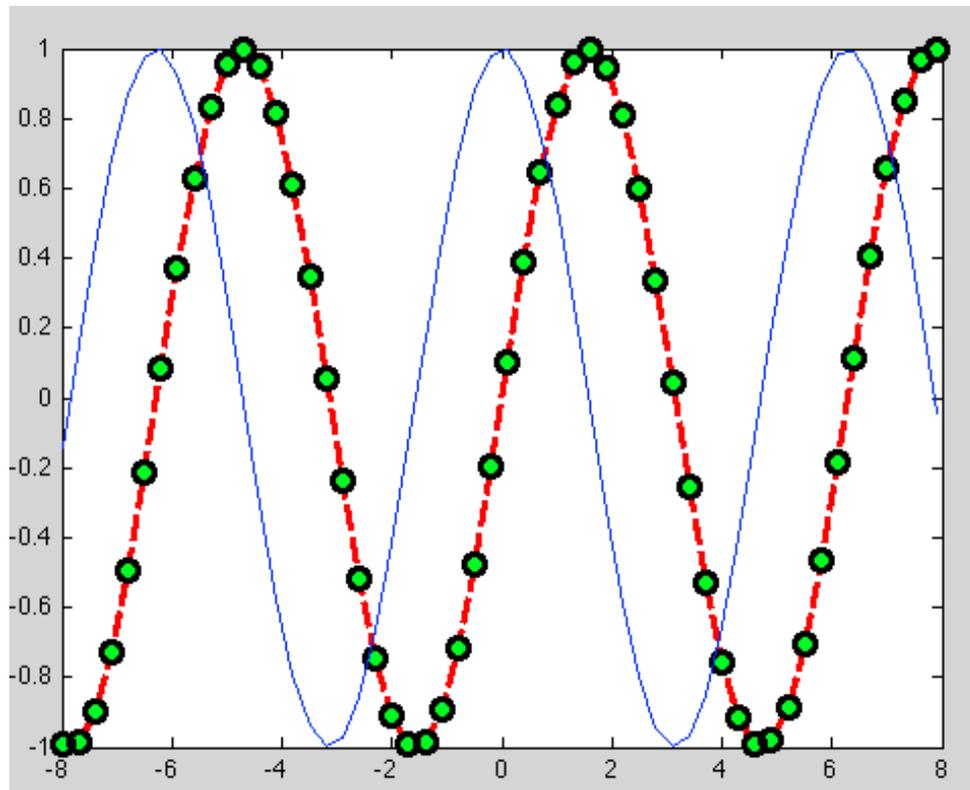


# Matlab: Creazione di grafici

Spesso dobbiamo aggiungere al grafico altre serie di dati per  
Avere tutto in un unico grafico:

`y1 = (cos(x)) ; %nuova serie`

`hold on` % aggiunge (al grafico precedente) la nuova serie di dati

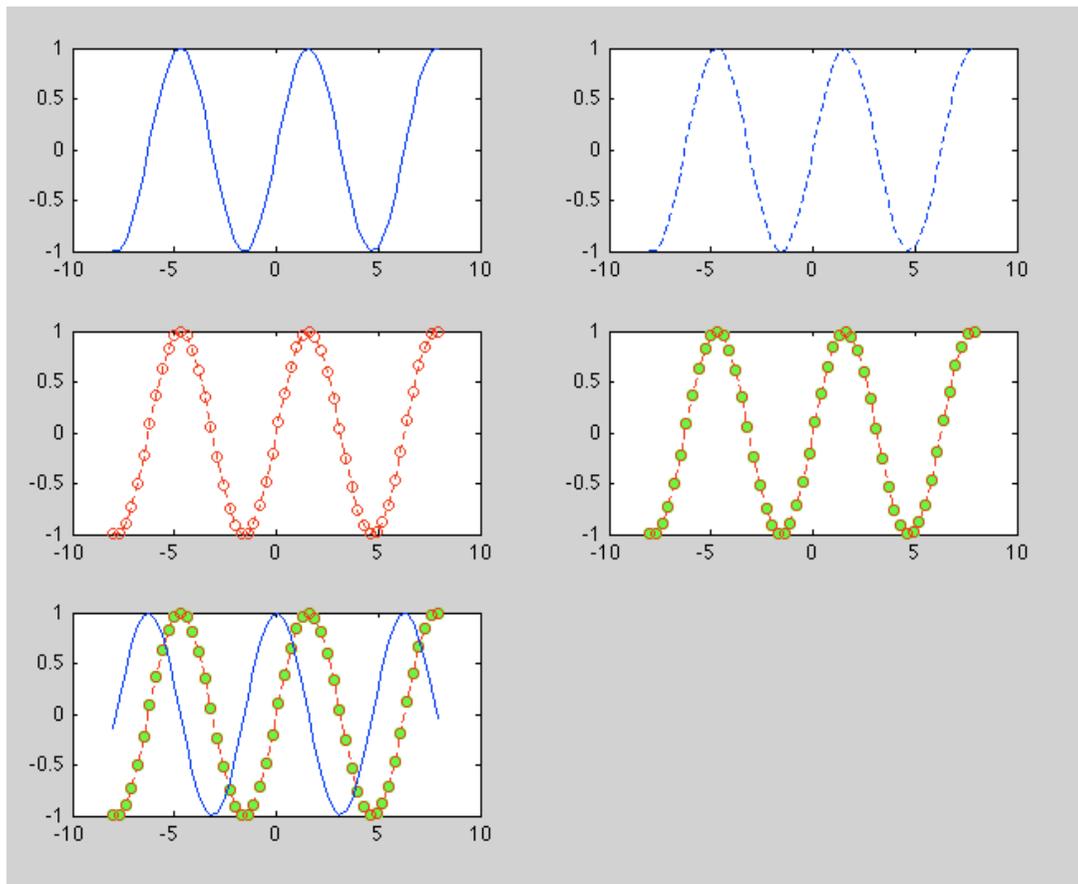


Subplot: una figura con più grafici  
subplot (r, c, p)

Numero righe        r

Numero colonne     c

Poisizione scelta   p



**subplot(3,2,1)**  
**plot(x,y)**

**subplot(3,2,2)**  
**plot(x,y,'--')**

**subplot(3,2,3)**  
**plot(x,y,'--or')**

**subplot(3,2,4)**  
**plot(x,y,'--or','LineWidth',**  
**1,...**

**'MarkerFaceColor','g',...**  
**'MarkerSize',5)**

**subplot(3,2,5)**  
**plot(x,y,'--or','LineWidth',**  
**1,...**

**'MarkerFaceColor','g',...**  
**'MarkerSize',5)**

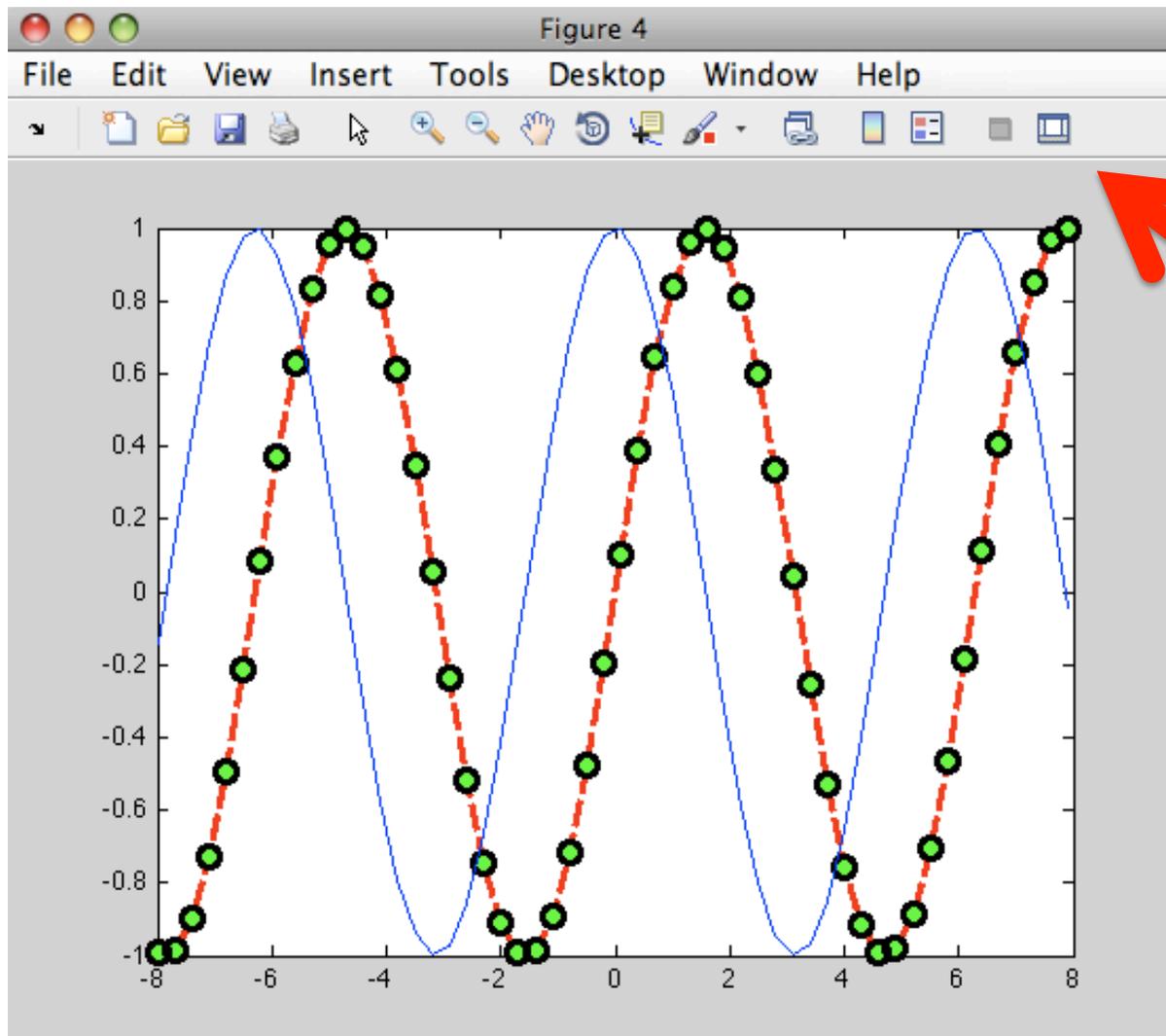
## Attenzione...

Se passiamo delle matrici

```
plot(x_matrice, y_matrice)
```

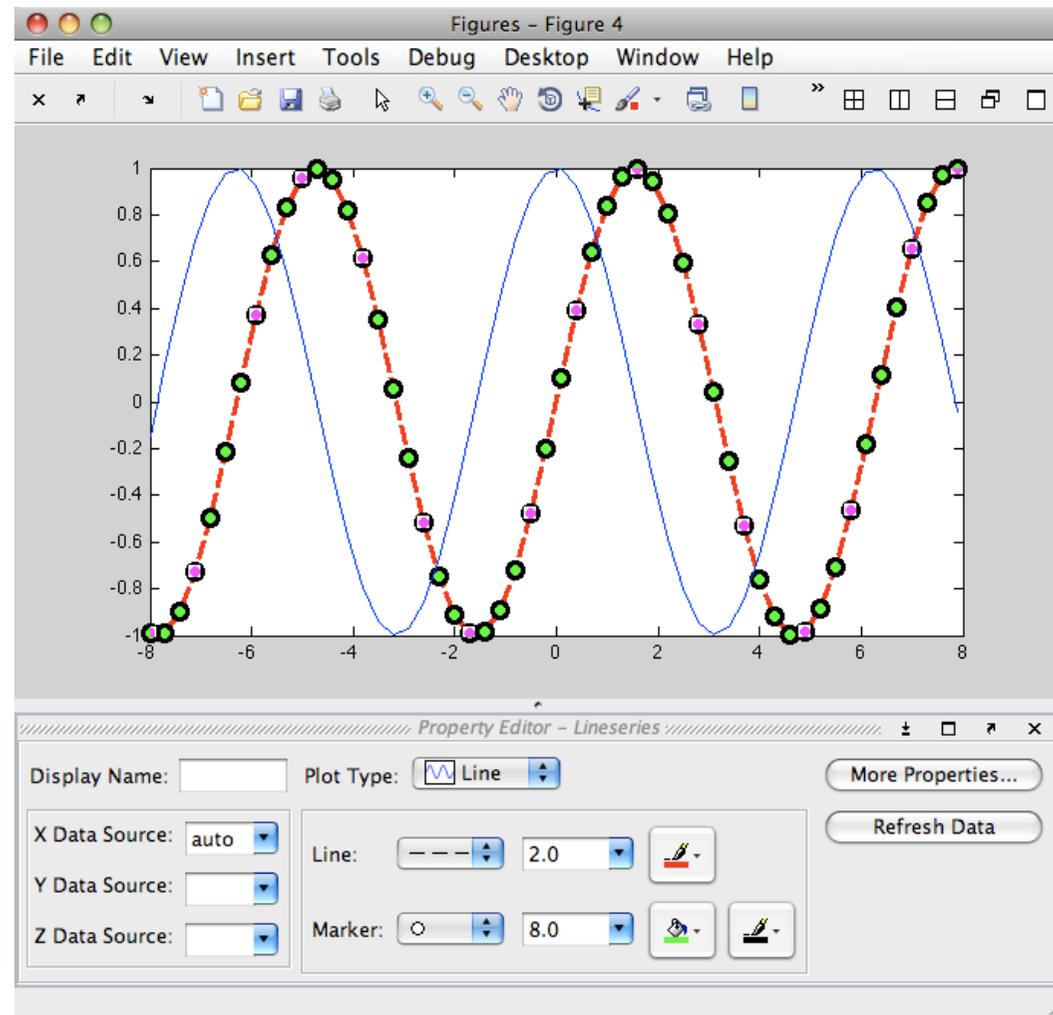
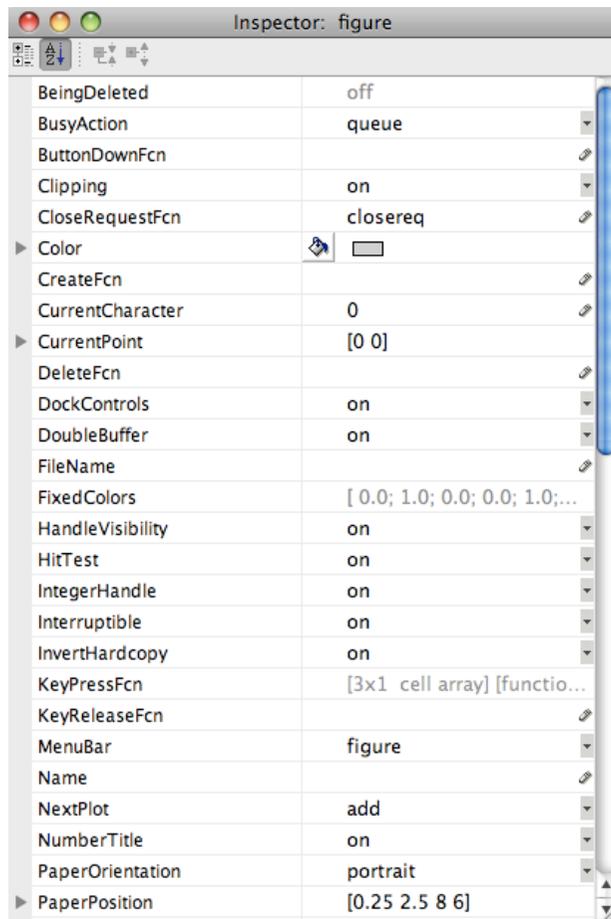
- Grafico colonna per colonna
- Cambia i colori ogni volta

I grafici possono anche essere modificati direttamente dalla finestra di dialogo del grafico stesso.



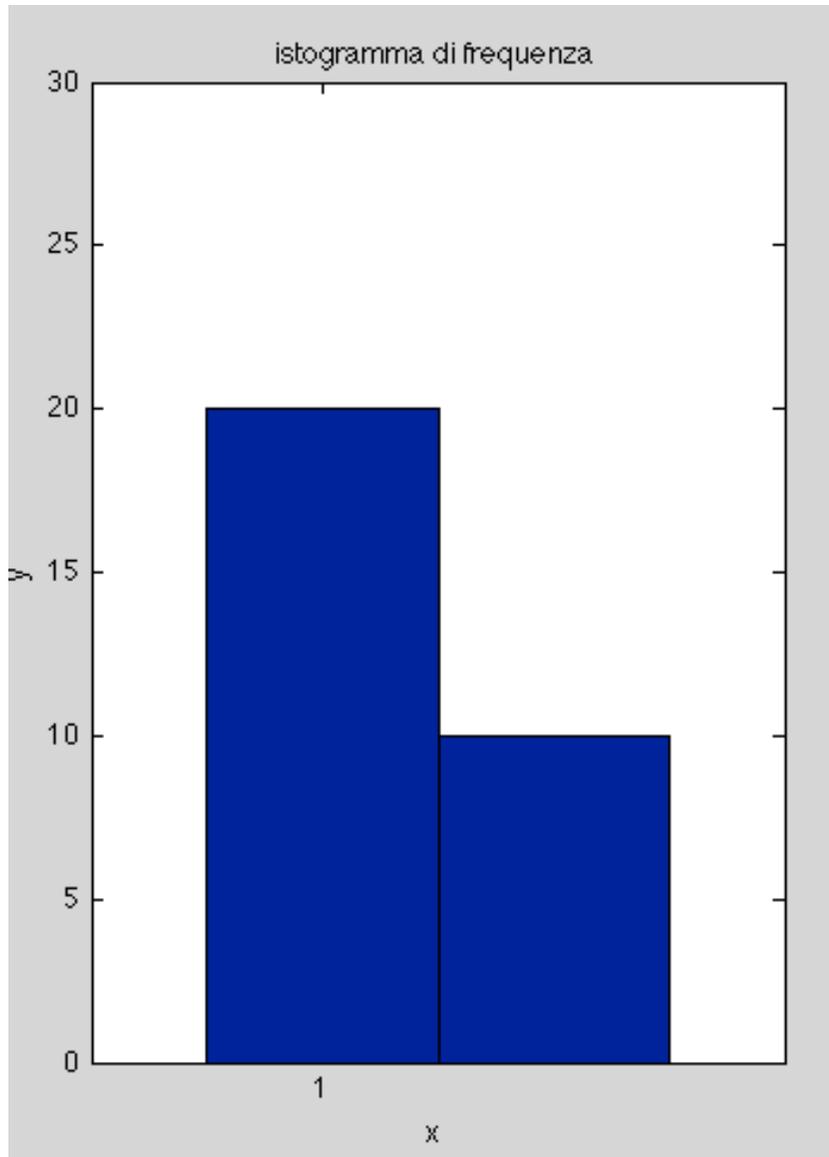
Però questo significa dover ripetere il procedimento ogni volta!!!!!!.

E' molto meno noioso avere già il codice che ci restituisce la figura come la desideriamo.



# Matlab: Creazione di grafici

## Istogrammi



```
xx=1;
```

```
yy=2;
```

```
var1=repmat(1,1,20);
```

```
var2=repmat(2,1,10);
```

```
hist(var1,xx)
```

```
hold on
```

```
hist(var2,yy)
```

```
axis([0 3 0 30])
```

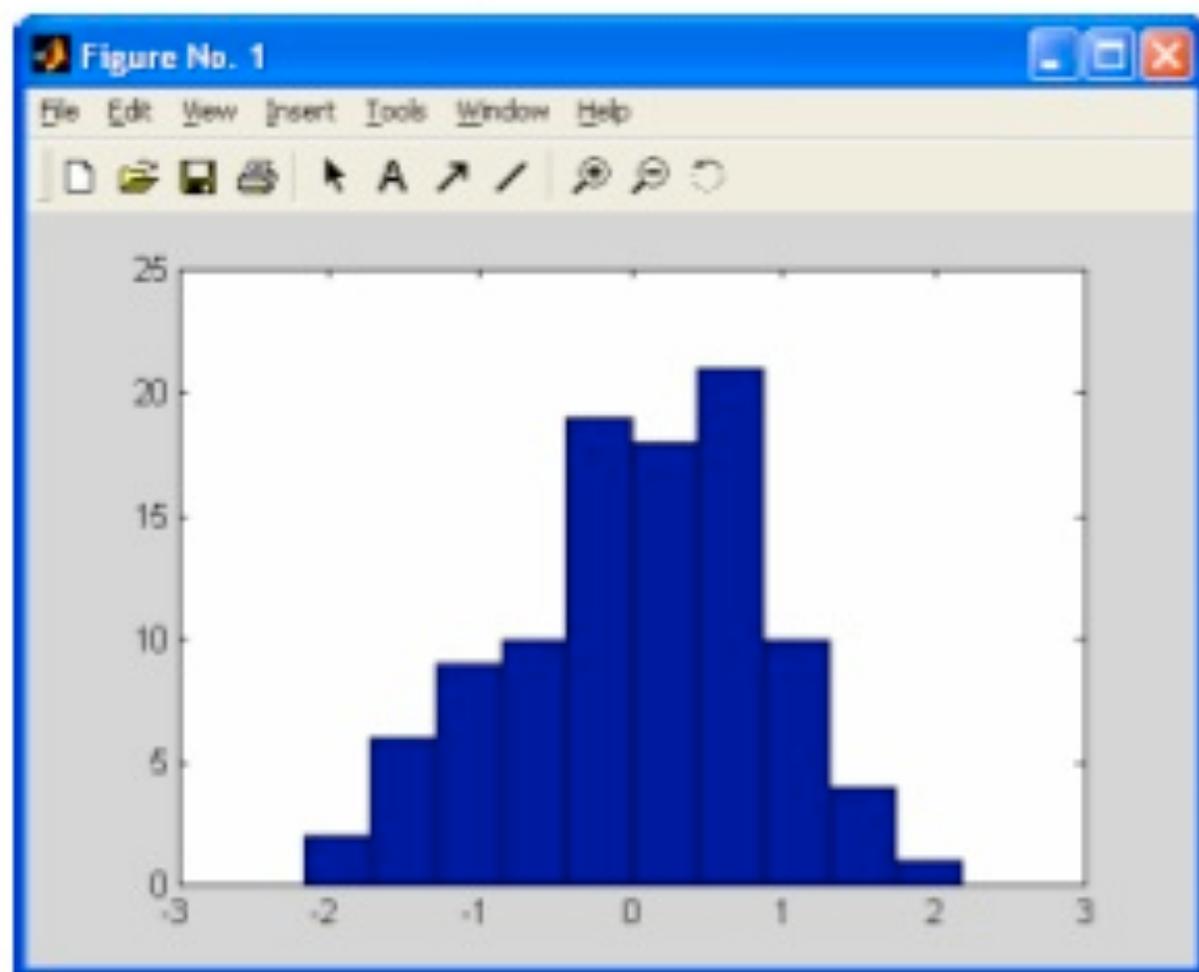
```
title('istogramma di frequenza')
```

```
xlabel('x');
```

```
ylabel('y');
```

- Istogrammi:

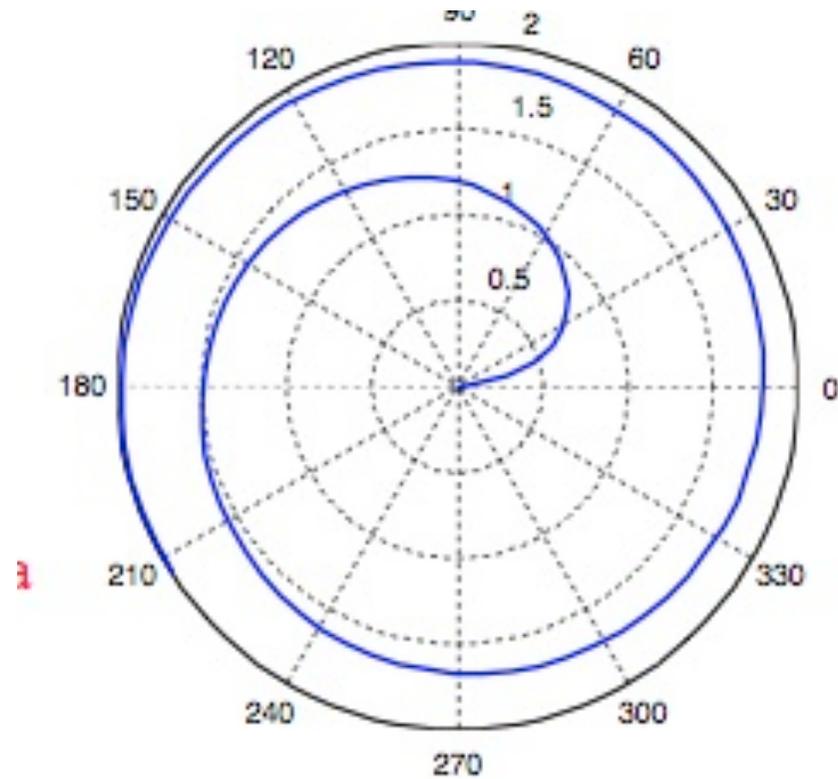
```
x = randn(1,100);  
hist(x)
```



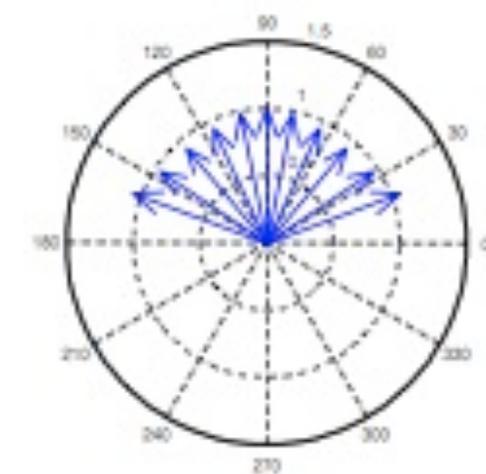
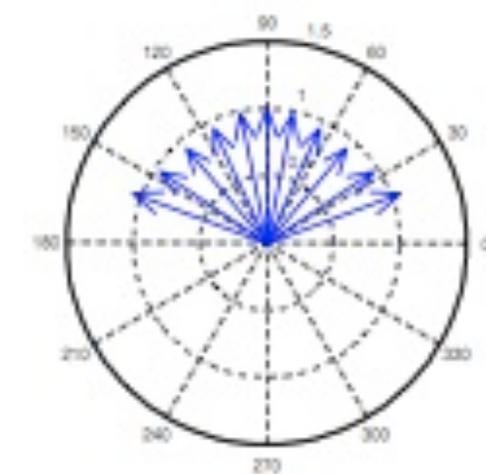
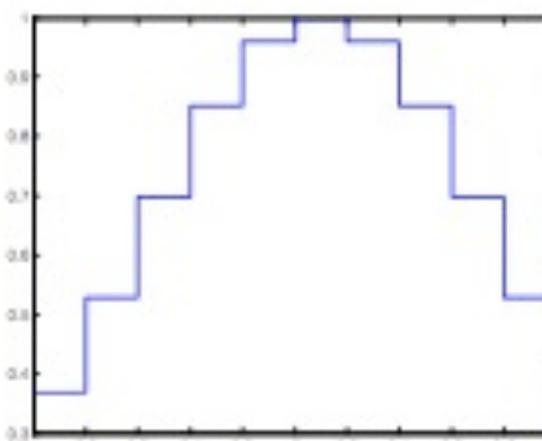
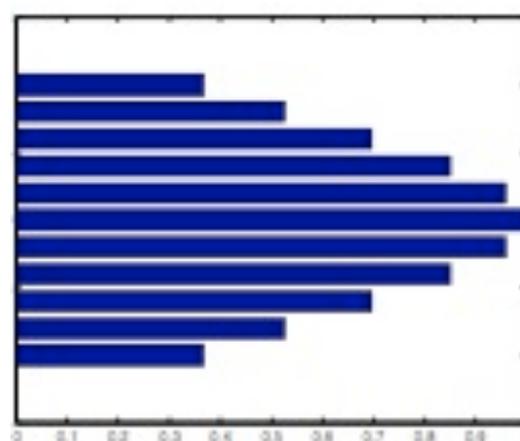
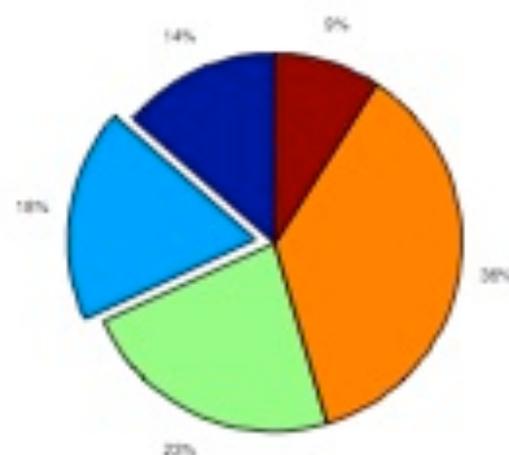
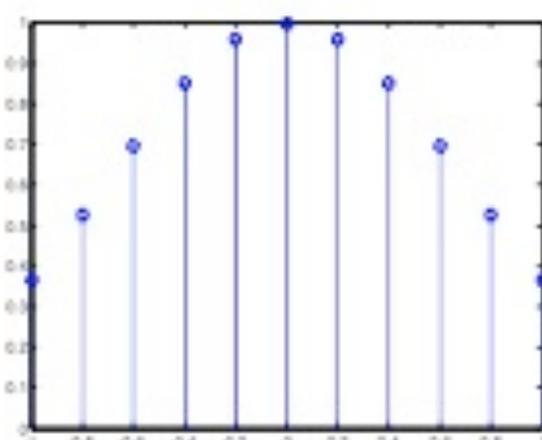
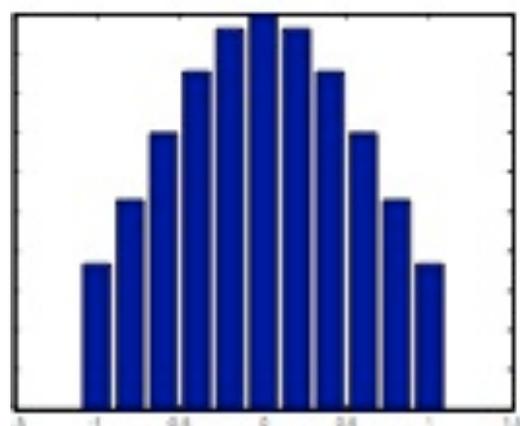
# Grafici 2-D Polari

```
x=1:100;  
r=log10(x)  
t=x/10
```

```
Polar(t,r)
```

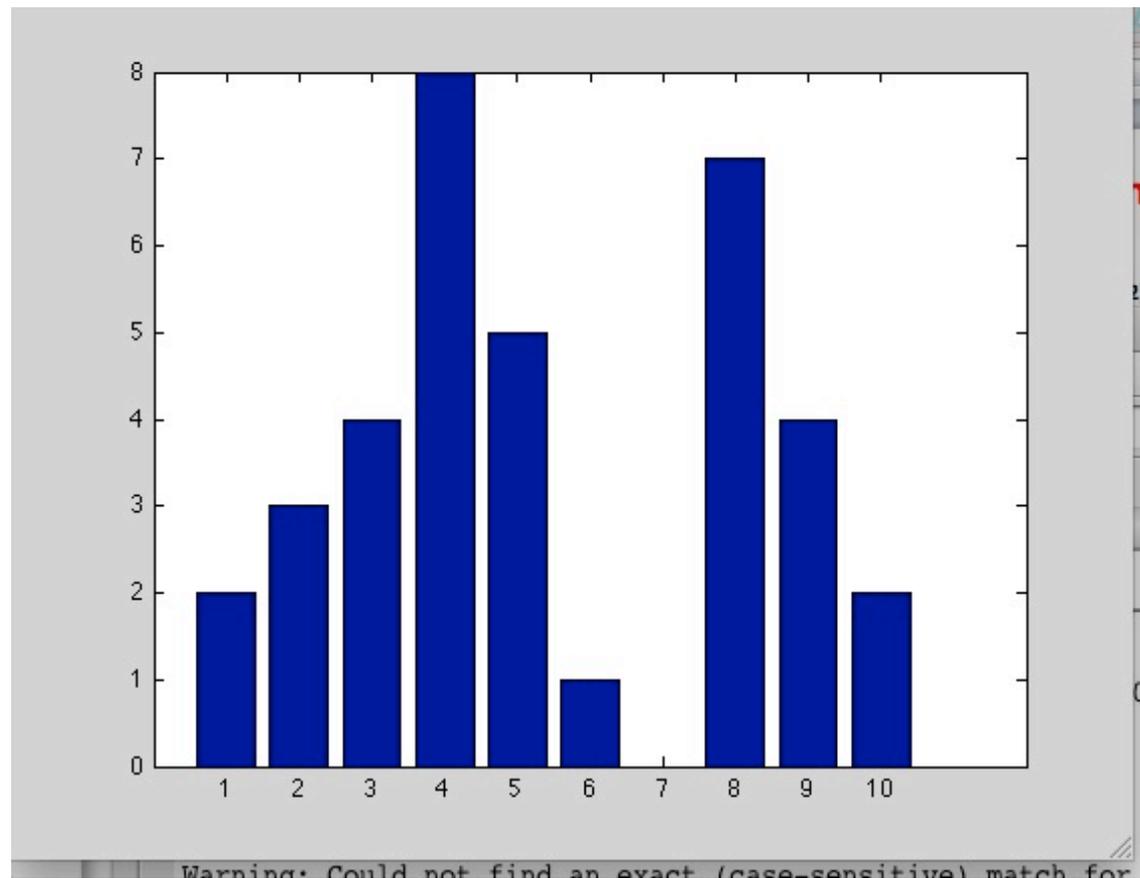


- Vertical and horizontal **bar** plots, **stem** and **stair** plots, **pie** and **compass** plots:



# Funzione Bar : Grafici a Barre

```
bar(1:10,[2,3,4,8,5,1,0,7,4,2])
```

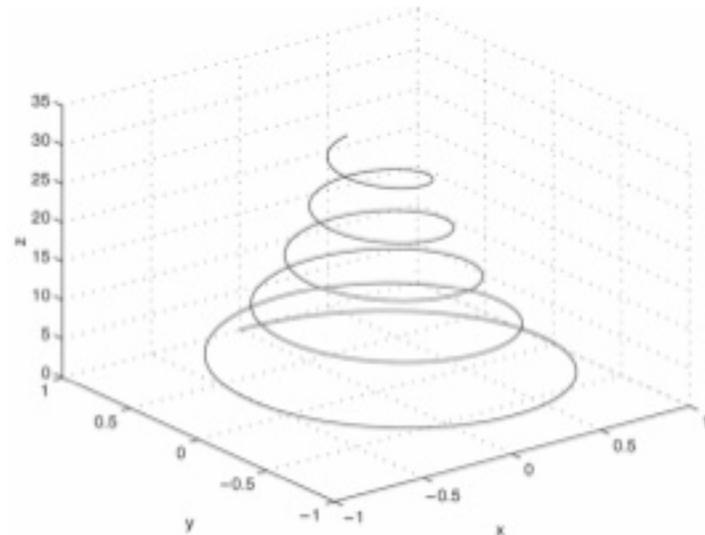


Warning: Could not find an exact (case-sensitive) match for

## Grafici 3-D

```
>>t = [0:pi/50:10*pi];  
>>plot3(exp(-0.05*t).*sin(t),...  
exp(-0.05*t).*cos(t),t),...  
xlabel('x'),ylabel('y'),zlabel('z'),grid
```

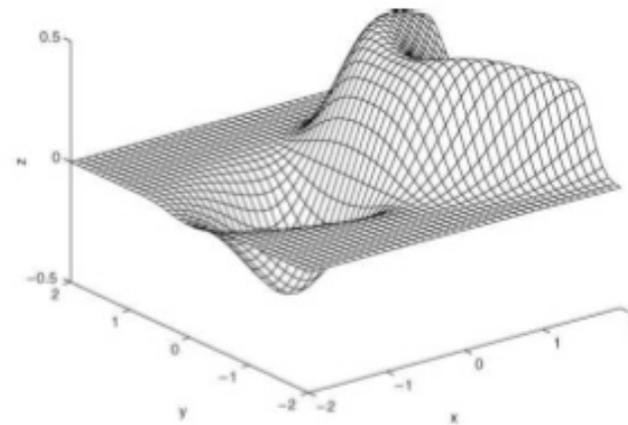
La curva  $x = e^{-0.05t} \sin t$ ,  $y = e^{-0.05t} \cos t$ ,  $z = t$ ,  
disegnata con plot3



# Superfici

```
>>[X,Y] = meshgrid(-2:0.1:2);  
>>Z = X.*exp(-((X-Y.^2).^2+Y.^2));  
>>mesh(X,Y,Z),xlabel('x'),ylabel('y'),...  
zlabel('z')
```

Funzione  $z = xe^{-[(x-y^2)^2+y^2]}$



# Cenni di statistica

## Misure di posizione e di dispersione

Istruzioni `mean`, `median`, `var`, `std`

**`mean(X)`** se **X** è un vettore restituisce il valore medio (media aritmetica) degli elementi di **X**;

**`median(X)`** se **X** è un vettore restituisce la mediana degli elementi in **X**;

**`var(X)`** se **X** è un vettore restituisce la varianza di **X**;

**`std(X)`** se **X** è un vettore restituisce la deviazione standard di **X**  
( **`std(X) = sqrt(var(X))`** );

# Cenni di statistica

## Distribuzione normale (1)

Istruzione `normpdf`

**`Y = normpdf(X, mu, sigma)`** restituisce in **Y** la distribuzione normale con valore medio **mu** e deviazione standard **sigma** calcolata in corrispondenza agli elementi di **X**

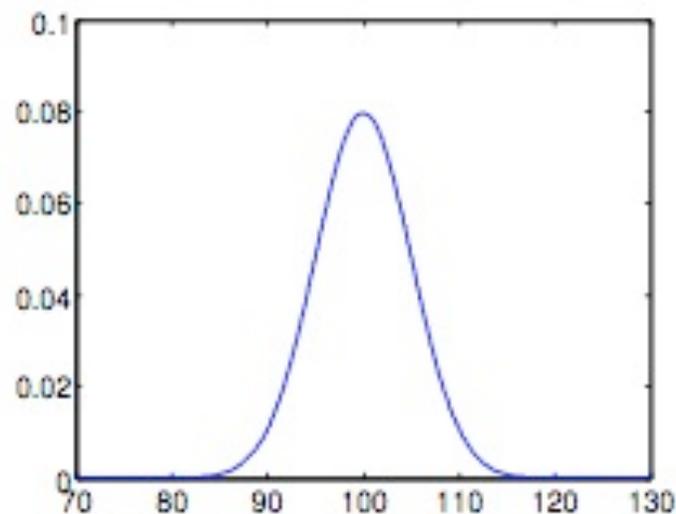
# Cenni di statistica

## Distribuzione normale (2)

Si considerino, ad esempio, le seguenti istruzioni:

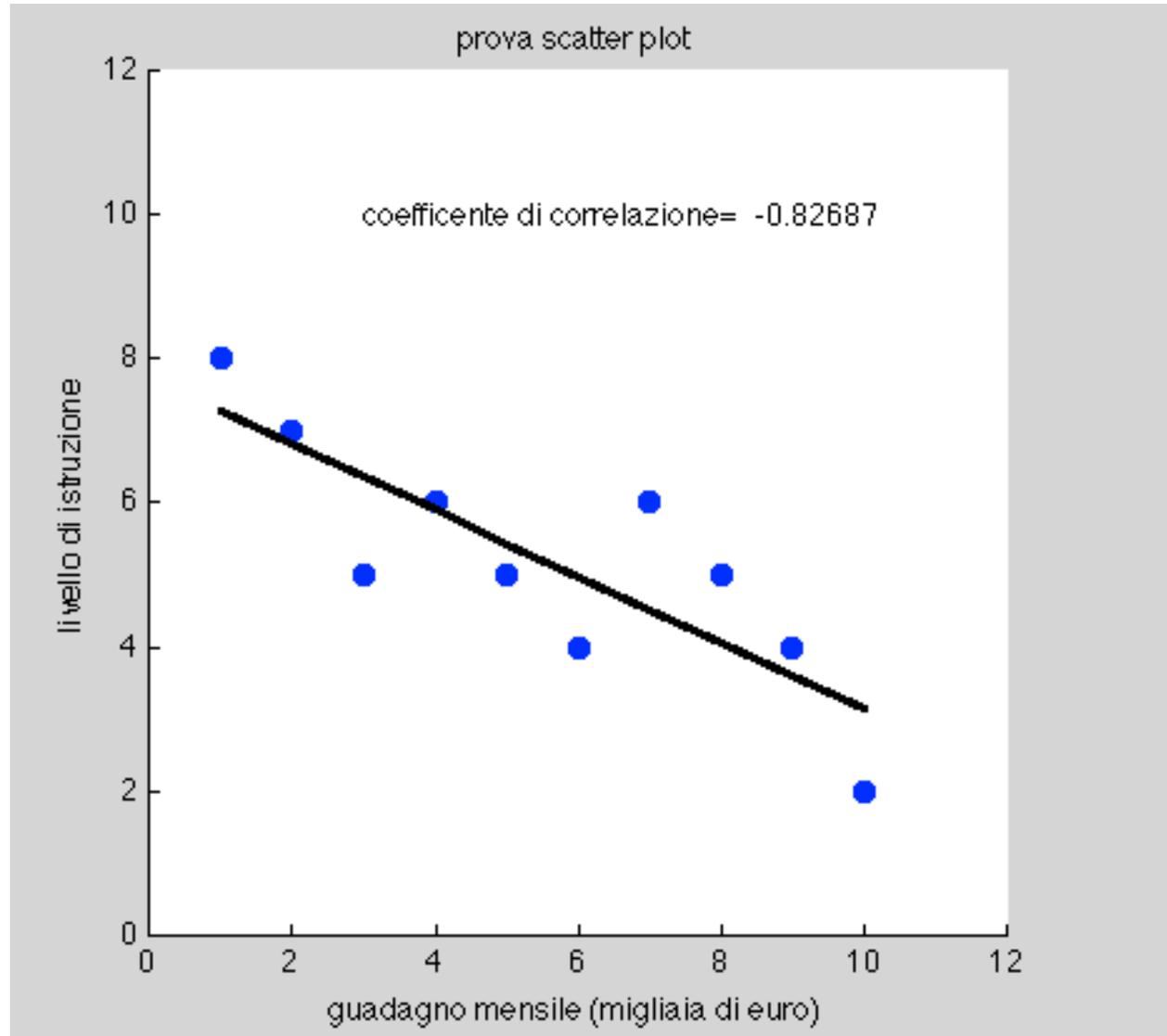
```
>> mu=100;  
>> sigma=5;  
>> x=[70:0.01:130];  
>> y=normpdf(x,mu,sigma);  
>> plot(x,y)
```

Esse generano la seguente figura:



# Matlab: Creazione di grafici

## Scatter plot

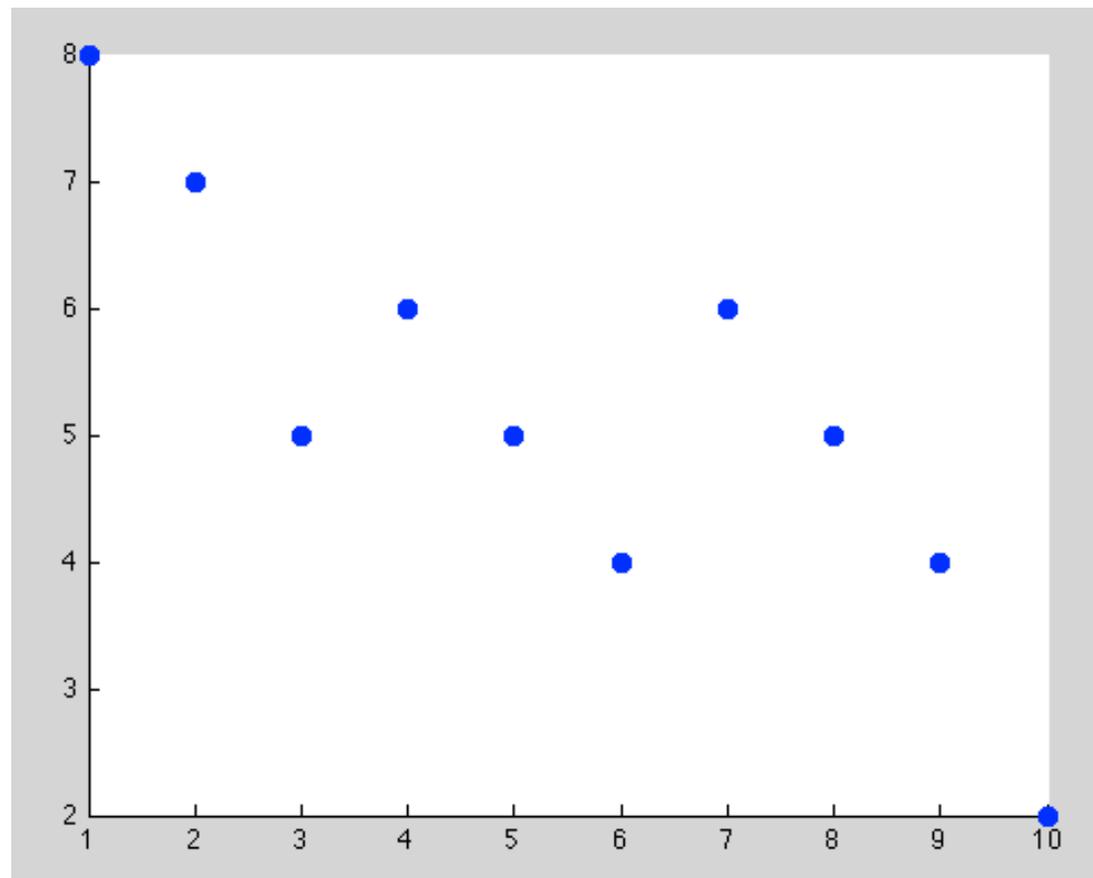


# Matlab: Creazione di grafici

```
x=(1:1:10);  
y=[8,7,5,6,5,4,6,5,4,2];
```

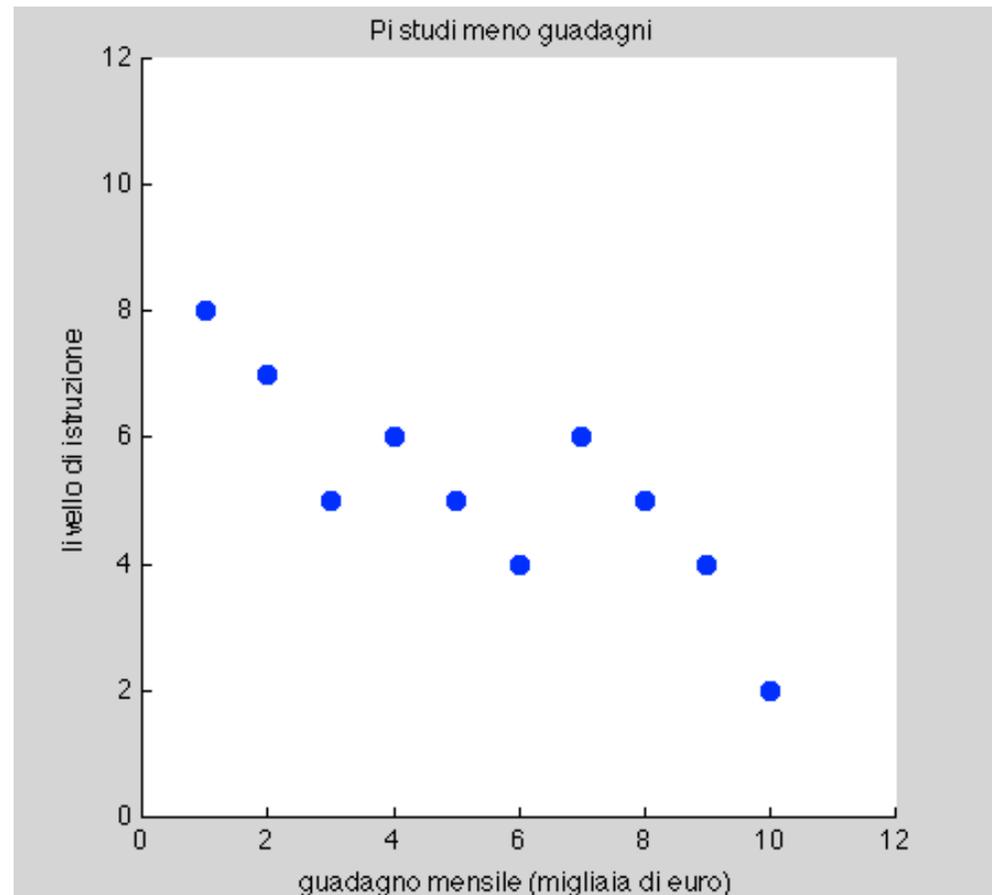
Come per il plot, cerchiamo i parametri di modifica nell' help !!!

```
scatter (x,y,50,'b','filled') %visualizzimole con uno  
scatter plot
```



# Matlab: Creazione di grafici

```
axis([0 12 0 12]) % definiamo limiti degli assi  
axis('square') % assi di uguale lunghezza  
xlabel('guadagno mensile (migliaia di euro)') % nome dell'asse x  
ylabel('livello di istruzione') % nome dell'asse y  
title('Più studi meno guadagni') % nome del grafico
```



# Matlab: Creazione di grafici

Misuriamo la correlazione tra le 2 variabili

`r=corrcoef(x,y);`  
correlazione lineare

`%calcola coeff di`

```
>>
>>
>>
>>
>>
>>
>> r
>> r
r =
    1.00    -0.83
   -0.83    1.00
>>
>>
>>
>>
>>
```

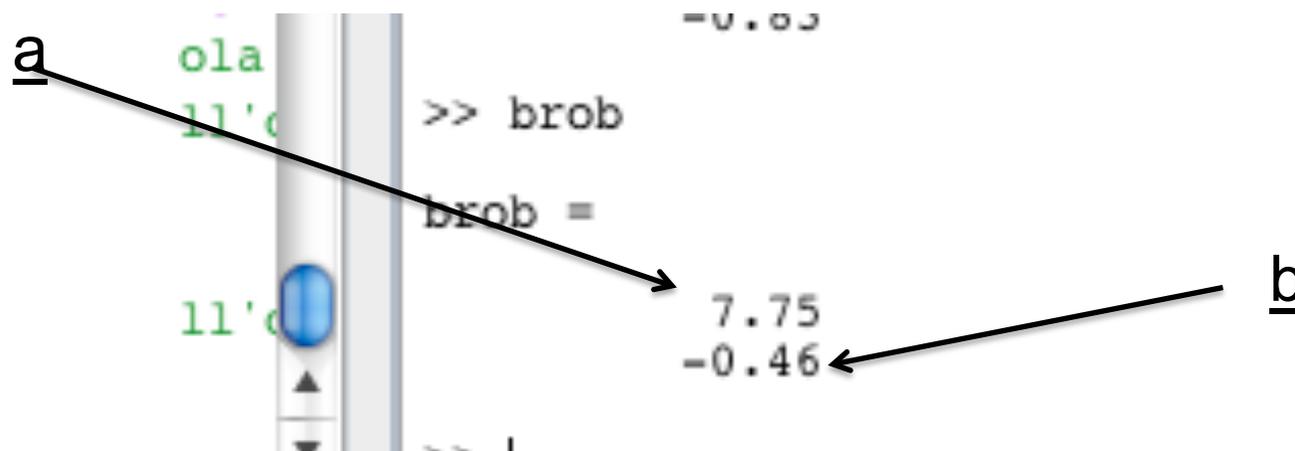
```
istru >>
+ gua >>
cola >> r(2);%lo estraiamo dall'output
all'o >> r(2)
ans =
-0.83
```

# Matlab: Creazione di grafici

brob = **robustfit**(x,y); % fit lineare

In "brob" ci sono salvati i coefficienti di regressione: a(intercetta) e b(coeff. angolare)

Fit lineare dei dati, 'robust' poiché poco influenzabile da dati 'anomali'



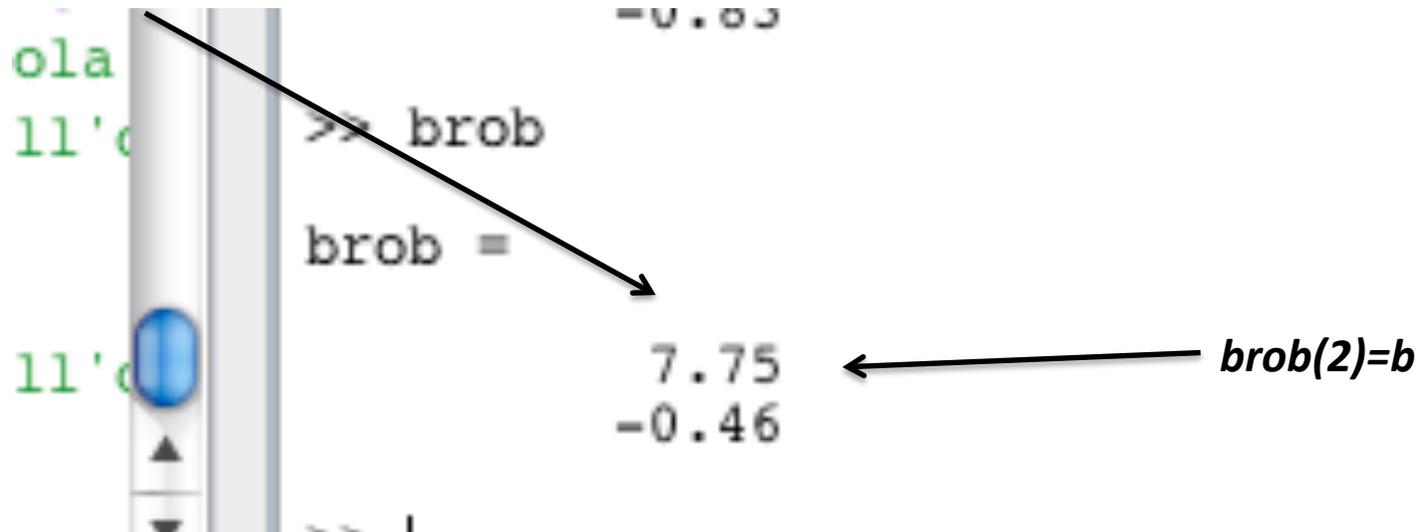
```
ola  
11'c  
11'c  
>> brob  
brob =  
    7.75  
   -0.46
```

# Matlab: Creazione di grafici

Inseriamo i parametri in una funzione lineare  
e  
chiediamogli di plottarla

`plot(x,brob(1)+brob(2)*x,'k-','linewidth',2);`

`brob(1)=a`



The screenshot shows a MATLAB command window with the following content:

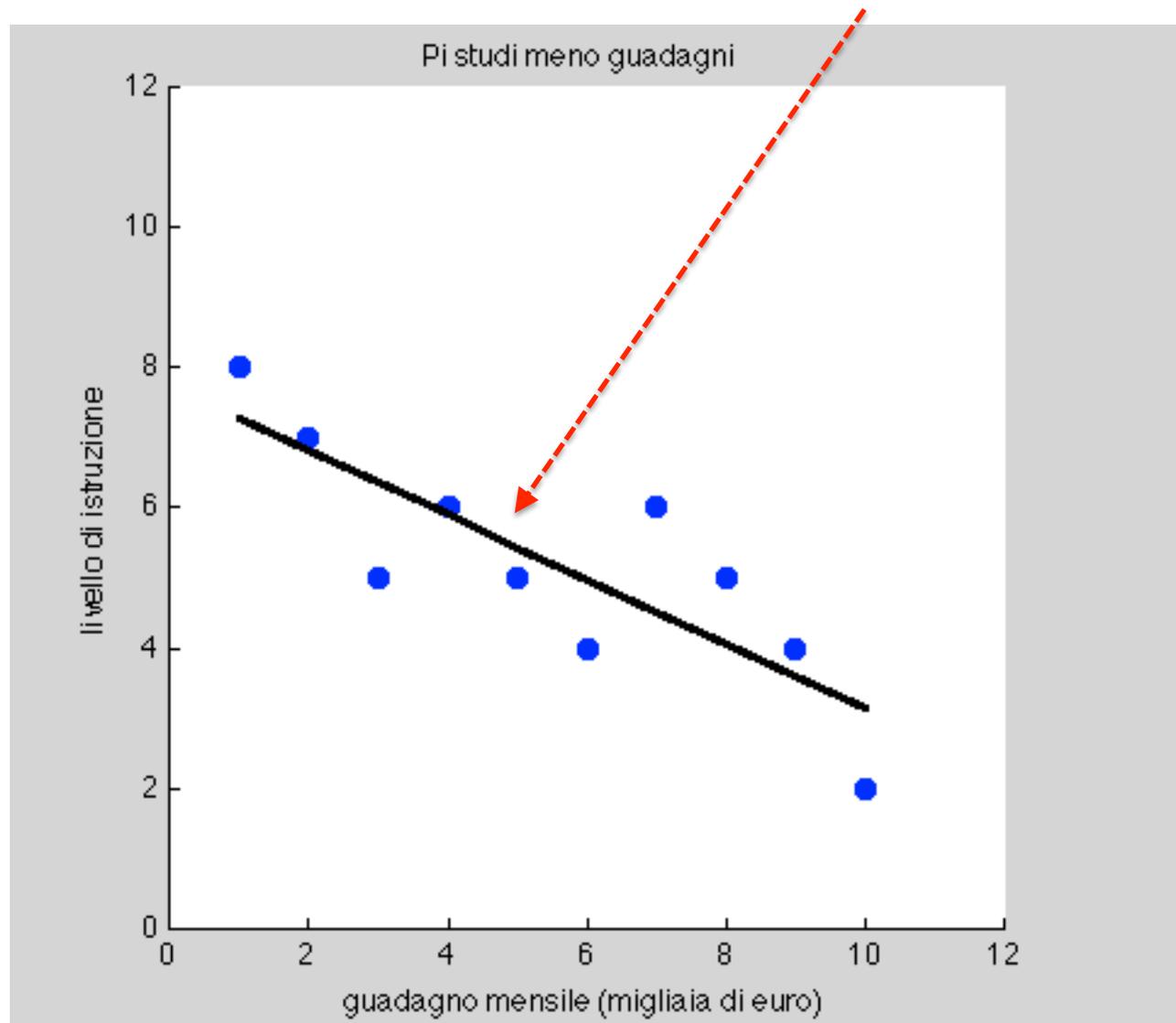
```
ola  
11'c >> brob  
brob =  
7.75  
-0.46
```

Annotations in the image include:

- An arrow pointing from the text "Inseriamo i parametri in una funzione lineare e chiediamogli di plottarla" to the `brob` variable in the code.
- An arrow pointing from the text "`brob(1)=a`" to the first element of the output vector, `7.75`.
- An arrow pointing from the text "`brob(2)=b`" to the second element of the output vector, `-0.46`.

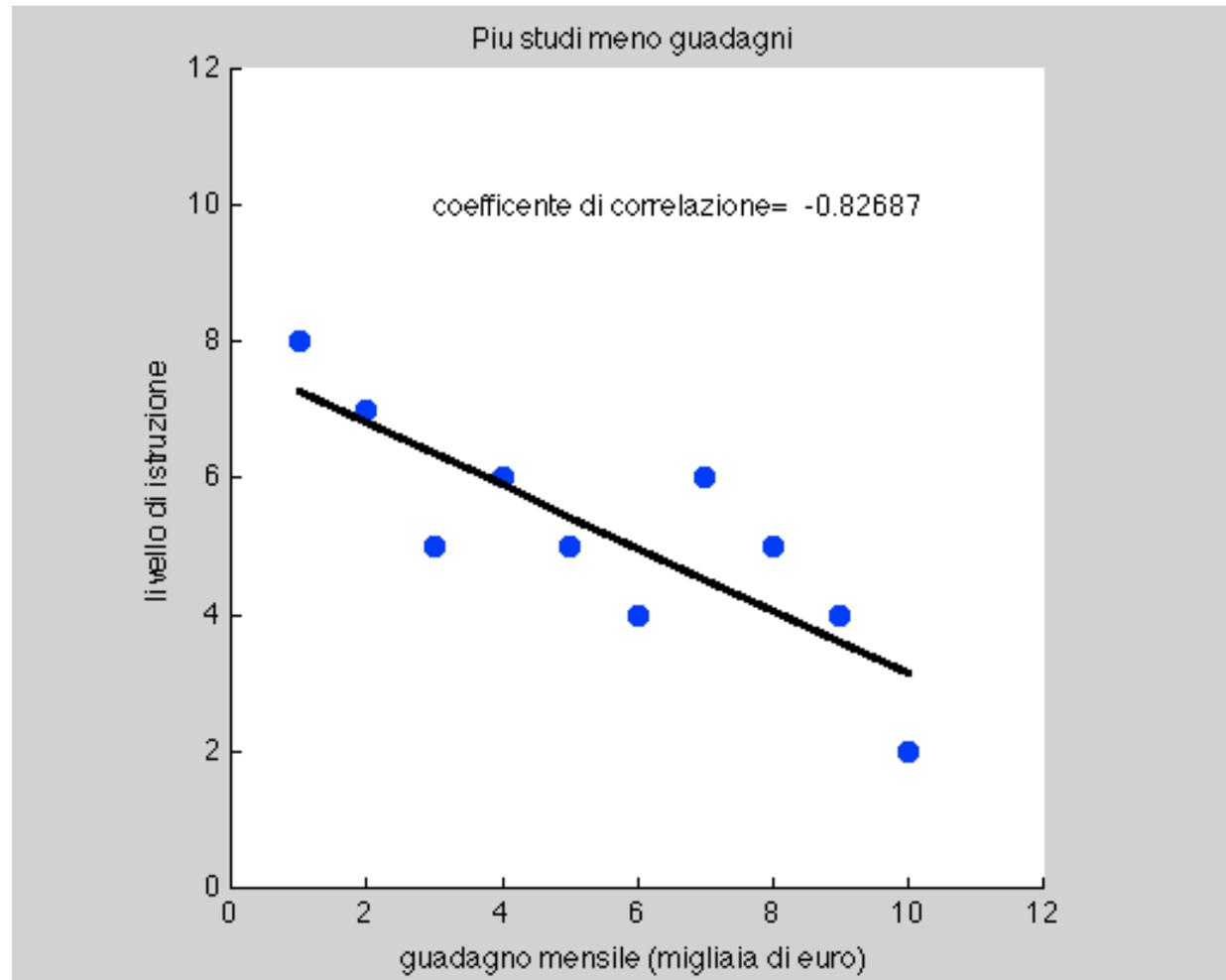
# Matlab: Creazione di grafici

```
plot(x,brob(1)+brob(2)*x,'k-', 'linewidth',2);
```



# Matlab: Creazione di grafici

```
corr=num2str(r(2)); %trasformiamolo in una stringa di testo  
text(3,10,'coefficiente di correlazione=') % coordinate dove vuoi scrivere  
text(8.5,10,corr)
```



# Grafica: Plot in Matlab

## Finestra riassuntiva

- *figure(N)* apre la finestra grafica N o la attiva se è già esistente
- *clf* ripulisce la pagina grafica attiva
- *hold* attiva e disattiva il mantenimento del grafico nella finestra grafica attiva
- *hold on, hold off* attiva (on) e disattiva (off) il mantenimento del grafico nella finestra grafica attiva
- *axis([xmin xmax ymin ymax])* per modificare i limiti degli assi cartesiani
- *title('titolo')* specifica il testo da porre sopra il grafico
- *xlabel('testo')* specifica il testo da porre sull'asse x
- *ylabel('testo')* specifica il testo da porre sull'asse y
- *text(x,y,'testo')* specifica il testo da porre nella posizione x,y del grafico

# Matlab: Interagire con l'OS

Il Workspace: Comprendere, utilizzare ed interrogare gli elementi di lavoro

Tutte le variabili create vanno ad occupare uno spazio nella memoria della macchina e vengono mostrate nel workspace

Per interrogare il workspace si possono usare i comandi “who” o “whos”

Per togliere una variabile dalla memoria e farla cessare di esistere si deve usare il comando “clear”

**Ricordatevi che quando un programma è formato da dei sotto moduli (struttura NON a lista di spaghetti) le variabili all'interno di un modulo NON sono conosciute dal resto del sistema**

# Matlab: Interagire con l'OS

## Salvataggio e richiamo di variabili al di fuori della sessione di lavoro: load and save

Le variabili presenti nel workspace possono essere salvate dalla “memoria a breve termine” della macchina al suo hard disk in modo da averle disponibili anche per successive sessioni di lavoro.

**save “filename” nomevariabileasalvare**

Es:

```
save ('RisultatiMaria' , 'matriceRisultati');
```

Richiamo della variabile in una nuova sessione di lavoro:

**load “filename”**

Es:

```
load RisultatiMaria o 'RisultatiMaria' → Ci troveremo nel  
Workspace la variabile MatriceRisultati
```

# *Matlab: Interagire con l'OS*

## *Comandi per interagire con il sistema operativo*

What: Lista dei file .m nella directory corrente

Dir o ls: Lista di tutti i file nella directory corrente

cd <path>: Cambia la directory di lavoro

Pwd: Mostra la directory corrente di lavoro

Which: Mostra tutto il path di un file .m

Computer: Tipo di computer

# Salvataggio e Richiamo Dati

Per salvare i nomi e i valori delle variabili create durante una sessione di Matlab si può utilizzare il comando ***save***

## **Save filename variabili**

in questo caso le variabili vengono salvate in un file .mat e cioè in un formato binario leggibile solo da Matlab.

## **Save filename variabili-ascii**

In questo caso i valori delle variabili vengono salvate in un file ascii e cioè un formato American Standard Code for information interchange riconosciuto da tutti i word processor.

Per importare le variabili memorizzate in file si utilizza il comando ***LOAD***

## Creo e salvo i dati

```
disp('introduci la temperatura media in gradi centigradi della stanza ora  
per ora')  
fori=1:24  
disp(['ora ',num2str(i)]);  
x(i)=input('introduci temperatura media = ');  
end  
save temperature x;
```

## Carico ed estraggo i dati

```
load temperature  
n=input('di che ora vuoi sapere la temperatura media? ');  
disp( ['si aveva una temperatura media di gradi ',num2str(x(n))]);
```

## Salvare e importare dati da Excel

Per salvare, in un file excel, i valori della variabili create durante una sessione di Matlab si può utilizzare il comando *xlswrite*.

**`xlswrite('filename', M)`**

Scrive la matrice *M* nel file Excel. La matrice in ingresso *M* deve Essere numerica o di caratteri. Viene salvata nel primo foglio dalla cella A1.

Per leggere da un file Excel:

**`num= xlsread('filename')`**

Carica, in forma numerica, i valori presenti sul primo foglio excel e li attribuirà alla variabile *num*

```
disp('introduci la temperatura media in gradi centigradi della stanza ora  
per ora')  
fori=1:12  
disp(['ora ' num2str(i)]);  
x(i)=input('introduci temperatura media = ');  
End  
xlswrite('tempexcel', x)
```

```
k=xlsread('tempexcel')  
n=input('di che ora vuoi sapere la temperatura media? ');  
disp( ['si aveva una T di gradi ' num2str(k(n))]);
```