SIS Scuola Interdipartimentale delle **Scienze**, dell'**Ingegneria** e della **Salute**

DiST Dipartimento di Scienze e Tecnologie

Laurea Magistrale in STN

Applicazioni di Calcolo Scientifico e Laboratorio di ACS

(12 cfu)

prof. Mariarosaria Rizzardi

Centro Direzionale di Napoli – Isola C4
studio: n. 423 – Lato Nord, 4° piano
Tel.: 081 547 6545
email: mariarosaria.rizzardi@uniparthenope.it

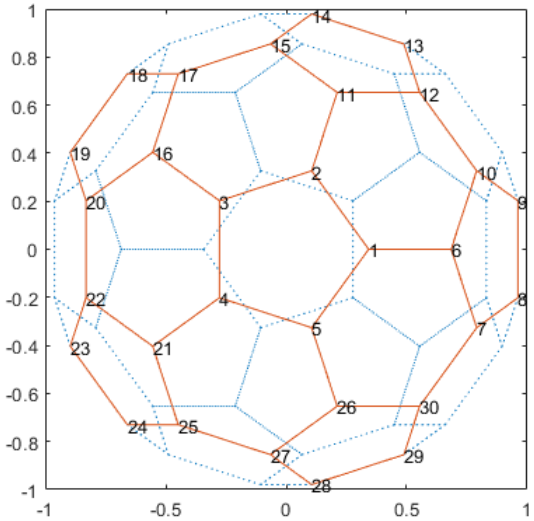
Argomenti trattati

- Introduzione ai tipi di dati strutturati in MATLAB:
  - sparse matrix,
  - cell array,
  - structure,
  - categorical array,
  - table.

Tipo di dati strutturati: matrici sparse

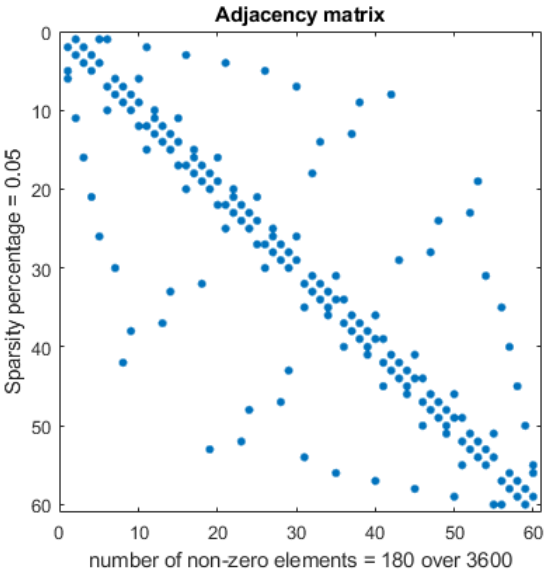
Una **matrice sparsa** è una matrice numerica o logica in cui solo **pochi elementi sono diversi da zero**.

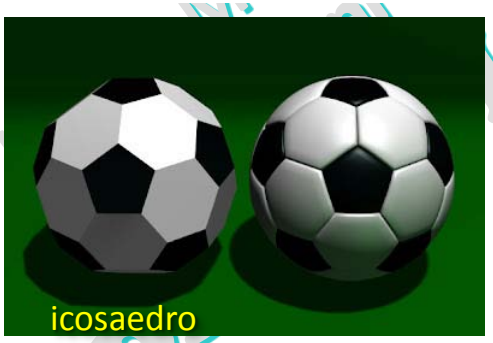
Esempio: bucky ball



Cupola geodetica di R. Buckminster Fuller di dimensione 60×60.

```
[A,V]=bucky; gplot(A,V,':')
axis('square'); hold on
gplot(A(1:30,1:30),V)
for k=1:30
  text(V(k,1),V(k,2),num2str(k))
end
figure; spy(A)
```





20 facce

```
Sperc=nnz(A)/numel(A);
fprintf("\nSparsity percentage = %g\n", Sperc)
Sparsity percentage = 0.05
Afull=full(A);
whos
```

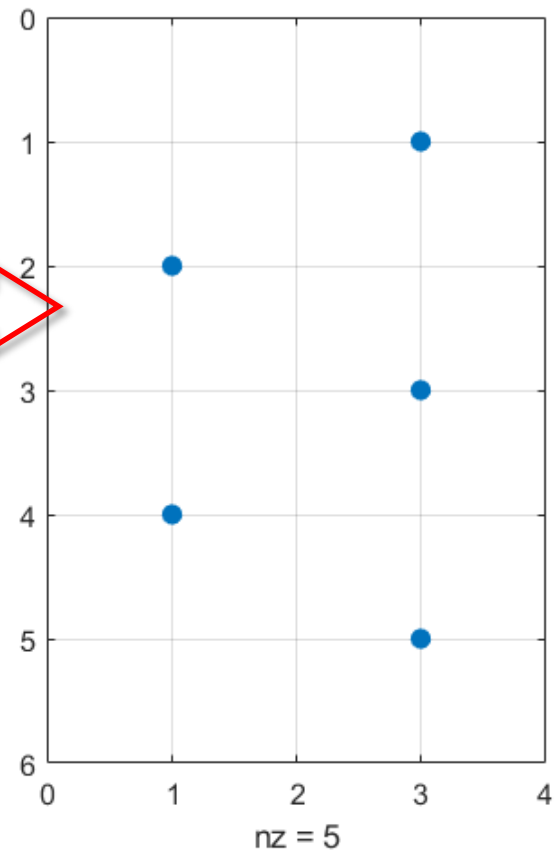
Name	Size	Bytes	Class	Attributes
A	60x60	3368	double	sparse
Afull	60x60	28800	double	

Esempio: matrice sparsa in MATLAB

```
A=[0 1.2 0 2.5 0;zeros(1,5);3.4 0 4.1 0 5.3]'
A =
         0         0    3.4
    1.2         0         0
         0         0    4.1
    2.5         0         0
         0         0    5.3
spy(A,24); grid on
S=sparse(A)
S =
   (2,1)        1.2
   (4,1)        2.5
   (1,3)        3.4
   (3,3)        4.1
   (5,3)        5.3
whos
Name        Size                Bytes  Class     Attributes

A           5x3                   120  double
S           5x3                   112  double    sparse
```

```
disp(full(S))
         0         0    3.4
    1.2         0         0
         0         0    4.1
    2.5         0         0
         0         0    5.3
```



num. elementi di A = 15

15 × 8 bytes =120 bytes (OK!)

in S num. elem. ≠ 0 = 5

5 × 8 bytes = 40 bytes ⇒? 112 bytes

Per memorizzare una matrice sparsa **S** di dimensioni **m×n**, con **nnz** elementi non nulli, **MATLAB** usa tre vettori (V, I_r, P_c):

- il primo (V) contiene gli elementi non nulli della matrice;
  lunghezza: **nnz**,
  tipo componente: numero reale-8bytes (o complesso-16bytes);
- il secondo (I_r) contiene per ciascun elemento non nullo il corrispondente indice di riga nella matrice;
  lunghezza: **nnz**,
  tipo componente: indice-puntatore-8bytes;
- il terzo (P_c) specifica l'inizio di ogni colonna della matrice nel vettore (V) degli elementi non nulli;
  lunghezza: **n+1**,
  tipo componente: indice-puntatore-8bytes.

formula occupazione di memoria (byte): **nnz***8 + **nnz***8 + (**n+1**)*8

Nell'esempio precedente, **[m,n]=size(S): m=5, n=3. nnz = 5**
nnz*8 + **nnz***8 + (**n+1**)*8 = 5*8 + 5*8 + 4*8 = 80 + 32 = 112 **(OK!)**

Nell'esempio …

indici di inizio delle colonne di A in **V**
+ numero elem ≠ 0 nella colonna

corrispondente indice di riga in A dell'elemento di **V**

elementi ≠0 in A

$$A_{(5\times 3)} = \begin{pmatrix} 0 & 0 & 3.4 \\ 1.2 & 0 & 0 \\ 0 & 0 & 4.1 \\ 2.5 & 0 & 0 \\ 0 & 0 & 5.3 \end{pmatrix}$$

n

P_c
1
3
3
6

La 1ª colonna di A inizia con l'elemento $V(P_c(1))$ e contiene $P_c(2) - P_c(1) = 2$ elementi non nulli.

La 2ª colonna di A inizia con l'elemento $V(P_c(2))$ e contiene $P_c(3) - P_c(2) = 0$ elementi non nulli (è ∅).

La 3ª colonna di A inizia con l'elemento $V(P_c(3))$ e contiene $P_c(4) - P_c(3) = 3$ elementi non nulli.

Costruzione di matrici sparse

$$A_{(5\times 3)} = \begin{pmatrix} 0 & 0 & 3.4 \\ 1.2 & 0 & 0 \\ 0 & 0 & 4.1 \\ 2.5 & 0 & 0 \\ 0 & 0 & 5.3 \end{pmatrix}$$

```
v=[1.2 2.5 3.4 4.1 5.3];
R=[2 4 1 3 5];   indici di riga
C=[1 1 3 3 3];   indici di colonna
S=sparse(R,C,v)
S =
   (2,1)        1.2
   (4,1)        2.5
   (1,3)        3.4
   (3,3)        4.1
   (5,3)        5.3
disp(full(S))
         0         0    3.4
    1.2000         0         0
         0         0    4.1
    2.5000         0         0
         0         0    5.3
```

Si specificano solo gli elementi non nulli e, per ciascuno di essi, si indica la sua posizione nella matrice attraverso i corrispondenti indice di riga ed indice di colonna.

Costruzione di matrici sparse strutturate

matrice tridiagonale

$$\begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix}$$

$$= \begin{pmatrix} -2 & & & & \\ & -2 & & & \\ & & -2 & & \\ & & & -2 & \\ & & & & -2 \end{pmatrix} + \begin{pmatrix} & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \\ & & & & \end{pmatrix} + \begin{pmatrix} & & & & \\ 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \end{pmatrix}$$

1

```
D=sparse(1:5,1:5,-2*ones(5,1));
disp(full(D))
    -2     0     0     0     0
     0    -2     0     0     0
     0     0    -2     0     0
     0     0     0    -2     0
     0     0     0     0    -2
E=sparse(1:4,2:5,ones(4,1),5,5);
disp(full(E))
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
     0     0     0     0     0
S=D + E + E';
disp(full(S))
    -2     1     0     0     0
     1    -2     1     0     0
     0     1    -2     1     0
     0     0     1    -2     1
     0     0     0     1    -2
```

Costruzione di matrici sparse strutturate

matrice a banda

$$\begin{pmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{pmatrix}$$

2

```
N=5; e=ones(N,1);
S=spdiags([e -2*e e],-1:1,N,N);
disp(full(S))
    -2     1     0     0     0
     1    -2     1     0     0
     0     1    -2     1     0
     0     0     1    -2     1
     0     0     0     1    -2
```

3

```
N=5; e=ones(N,1);
S=spdiags([e 2*e -4*e 3*e e],-2:2,N,N);
disp(full(S))
    -4     3     1     0     0
     2    -4     3     1     0
     1     2    -4     3     1
     0     1     2    -4     3
     0     0     1     2    -4
```

$$\begin{pmatrix} -4 & 3 & 1 & & \\ 2 & -4 & 3 & 1 & \\ 1 & 2 & -4 & 3 & 1 \\ & 1 & 2 & -4 & 3 \\ & & 1 & 2 & -4 \end{pmatrix}$$

Costruzione di matrici sparse strutturate

```
N=5; v=(1:N)';
S=spdiags(v,0,N,N);
disp(full(S))
```

```
     1     0     0     0     0
     0     2     0     0     0
     0     0     3     0     0
     0     0     0     4     0
     0     0     0     0     5
```

```
N=5; v=(1:N)';
S=spdiags(v,+1,N,N);
disp(full(S))
```

```
     0     2     0     0     0
     0     0     3     0     0
     0     0     0     4     0
     0     0     0     0     5
     0     0     0     0     0
```

+1

$-1, 0, +1, +2$ rispetto alla diagonale principale

```
N=5; v=(1:N)';
S=spdiags(v,+2,N,N);
disp(full(S))
```

```
     0     0     3     0     0
     0     0     0     4     0
     0     0     0     0     5
     0     0     0     0     0
     0     0     0     0     0
```

+2

```
N=5; v=(1:N)';
S=spdiags(v,-1,N,N);
disp(full(S))
```

```
     0     0     0     0     0
     1     0     0     0     0
     0     2     0     0     0
     0     0     3     0     0
     0     0     0     4     0
```

-1

Tipo di dati strutturati: cell array

Un **cell array**, i cui **elementi sono detti celle**, è una generalizzazione del tipo array perché ogni cella può contenere un qualsiasi tipo di dato. Per costruire gli array si usano [ ], per i cell array si usano { }.

Esempio

```
c={42; rand(5); "abcd"}
c =
  3×1 cell array
    {[      42]}  uno scalare
    {5×5 double}  una matrice
    {["abcd"  ]}  una stringa
```

```
disp(c{2})
    0.75774    0.70605    0.82346    0.43874    0.48976
    0.74313   0.031833    0.69483    0.38156    0.44559
    0.39223    0.27692     0.3171    0.76552    0.64631
    0.65548   0.046171    0.95022     0.7952    0.70936
    0.17119   0.097132   0.034446    0.18687    0.75469
```

matrice di random

```
c={42; {2+3i 'K'}; "abcd"}
c =
  3×1 cell array
    {[    42]}
    {1×2 cell}
    {["abcd"]}
```

```
disp(c{2})
      {[2 +          3i]}    {'K'}
```

Esempio d'uso dei cell array nei menu

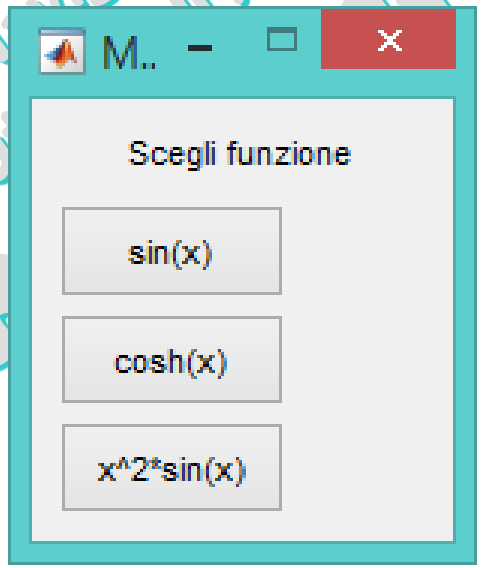
menù tramite cell-array di array di caratteri

```
C={'sin(x)','cosh(x)','x^2*sin(x)'};
MENU=menu('Scegli funzione',C)
```

oppure

menù tramite cell-array di stringhe

```
C={"sin(x)","cosh(x)","x^2*sin(x)"};
MENU=menu('Scegli funzione',C)
```



stesso menù tramite array … (matrice di caratteri)

tutte le righe dell'array devono avere lo stesso numero di caratteri

```
A=['sin(x)     '
   'cosh(x)    '
   'x^2*sin(x)'];
MENU=menu{'Scegli funzione',A(1,:),A(2,:),A(3,:)}
```

Tipo di dati strutturati: struct

Una **struct** (structure array) è un tipo di dati formato da contenitori detti **campi** (**fields**). Ogni campo può contenere qualsiasi tipo di dati. Si accede ai dati di un campo utilizzando la notazione: **structName.fieldName**

Esempio

```
student(1)=struct('nome','Bianchi Aldo','matr','123','num_es',5)
student =
  struct with fields:
      nome: 'Bianchi Aldo'
      matr: '123'
    num_es: 5
student(2)=struct('nome','Rossi Mario','matr','231','num_es',3)
student =
1 x 2 struct array with fields:
     nome
     matr
    num_es
disp(student(2).matr)
ans =
231
disp(max(student.num_es))
  5
```

I nomi dei campi possono contenere lettere (A-Z, a-z), cifre (0-9) e underscore; essi devono iniziare con una lettera. La lunghezza massima del nome di un campo è il valore di namelengthmax=63.

I cell array e le struct possono essere nidificati, nel senso che possono contenere al loro interno altri cell array e struct.

Tipo di dati strutturati: categorical array

Il "categorical array" è un tipo di dati per memorizzare dati con valori appartenenti a un insieme finito di categorie discrete. Il "categorical array" consente di memorizzare in modo efficiente e di manipolare dati non numerici, mantenendo i nomi significativi per i valori dei dati. Le categorie possono avere un ordinamento naturale, ma non è necessario.

```
C=categorical({'R','G','B','B','G','B'})
C =
  1×6 categorical array
     R      G      B      B      G      B
whos C
Name       Size              Bytes  Class
C          1x6                 326  categorical
C=categorical({'R','G','B','B','G','B'})'
C =
  6×1 categorical array
     R
     G
     B
     B
     G
     B
```

```
categories(C)
ans = 3×1 cell array
    {'B'}
    {'G'}
    {'R'}
C == 'B'
ans = 1×6 logical array
   0   0   1   1   0   1
C > 'B'
Error using  >  (line 30)
Relational comparisons are not allowed
for categorical arrays that are not
ordinal.
```

```
A=[1 3 2; 2 1 3; 3 1 2];
B=categorical(A,[1 2 3],{'red' 'green' 'blue'},'Ordinal',true)
B = 3x3 categorical
     red   blue green
     green red  blue
     blue  red  green
B > 'green'
ans = 3×3 logical array
   0   1   0
   0   0   1
   1   0   0
```

categoria ordinata: l'operatore **>** ora è definito!

```
C=categorical({'R','G','B','B','G','B'},'Ordinal',true);
C > 'B'
ans = 1×6 logical array
      1   1   0   0   1   0
```

Tipo di dati strutturati: categorical array

Il "categorical array" consente di memorizzare in modo efficiente dati non numerici.

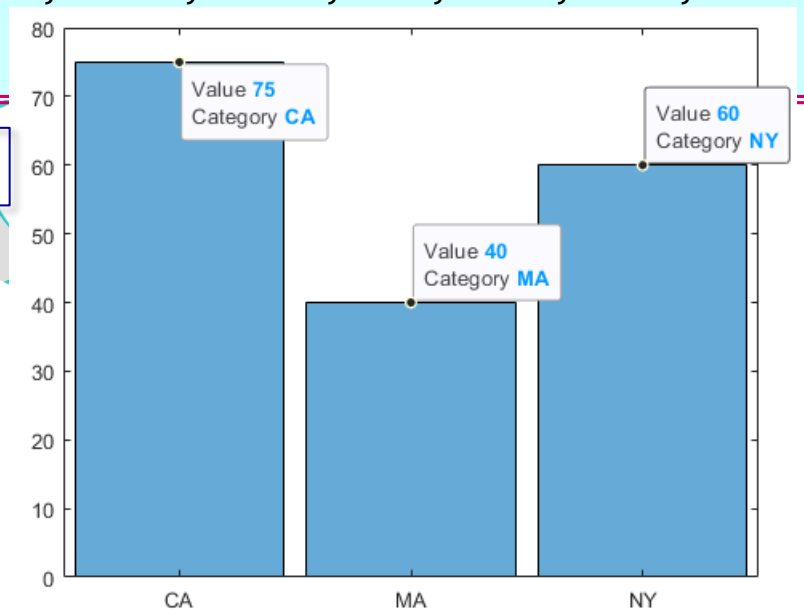
```
state1=[repmat({'MA'},25,1);repmat({'NY'},35,1);repmat({'CA'},75,1); ...
        repmat({'MA'},15,1);repmat({'NY'},25,1)];
whos state1
  Name          Size                Bytes  Class     Attributes
  state1      175x1                18900  cell
state2=categorical(state1);
whos state2
  Name          Size                Bytes  Class           Attributes
  state2      175x1                  501  categorical
histogram(state1)
Error using histogram: Expected input number 1, x, to be one of these types:
double, single, uint8, uint16, uint32, uint64, int8, int16, int32, int64, logical, datetime,
duration, categorical
histogram(state2)
```

crea un cell array

lo converte in un categorical array

occupa meno spazio

si può fare l'istogramma



Tipo di dati strutturati: table

La **table** è orientata ai dati tabellari spesso memorizzati in colonne (come in un foglio elettronico di Excel). Ogni variabile (colonna) di una tabella può essere di tipo diverso, con l'unica restrizione che tutte le variabili devono avere lo stesso numero di righe.

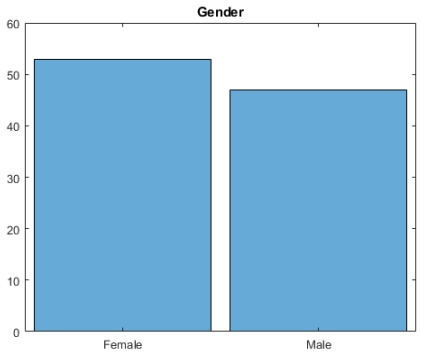
Esempio

```
load patients; whos
  Name                      Size              Bytes  Class
  Age                     100x1                 800  double
  Diastolic               100x1                 800  double
  Gender                  100x1               11412  cell
  Height                  100x1                 800  double
  …
  Smoker                  100x1                 100  logical
  Systolic                100x1                 800  double
  Weight                  100x1                 800  double
BloodPressure=[Systolic Diastolic];
T=table(Gender,Age,Smoker,BloodPressure);
disp(T(1:5,:))
    Gender       Age    Smoker    BloodPressure
    _________    ___    ______    _____________
    {'Male'  }    38    true       124      93
    {'Male'  }    43    false      109      77
    {'Female'}    38    false      125      83
    {'Female'}    40    false      117      75
    {'Female'}    49    false      122      80
```

```
disp(T{1:5,2})
    38
    43
    38
    40
    49
```

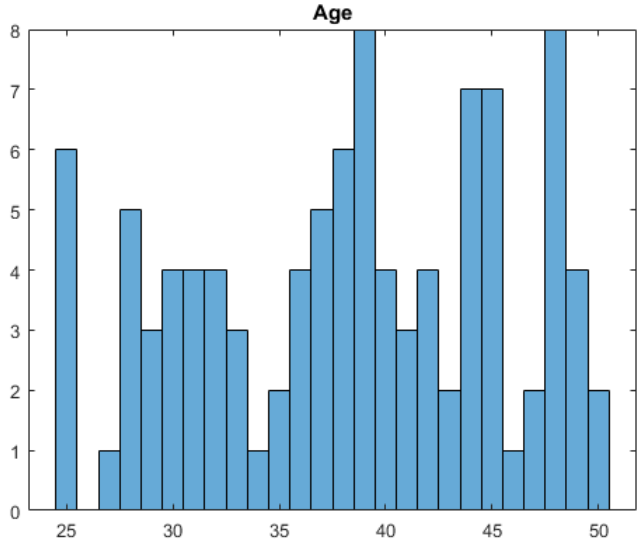
Tipo di dati strutturati: table

```
load patients
BloodPressure=[Systolic Diastolic];
T=table(Gender,Age,Smoker,BloodPressure);
disp(T.Properties)
 TableProperties with properties:
            Description: ''
               UserData: []
         DimensionNames: {'Row'  'Variables'}
          VariableNames: {'Gender'  'Age'  'Smoker'  'BloodPressure'}
    VariableDescriptions: {}
           VariableUnits: {}
      VariableContinuity: []
                RowNames: {}
disp(T.Properties.VariableNames)
 {'Gender'}  {'Age'}  {'Smoker'}  {'BloodPressure'}
disp(T.Properties.VariableNames{2})
Age
histogram(T{:,2})
title(T.Properties.VariableNames{2})
histogram(categorical(T{:,1}))
title(T.Properties.VariableNames{1})
```



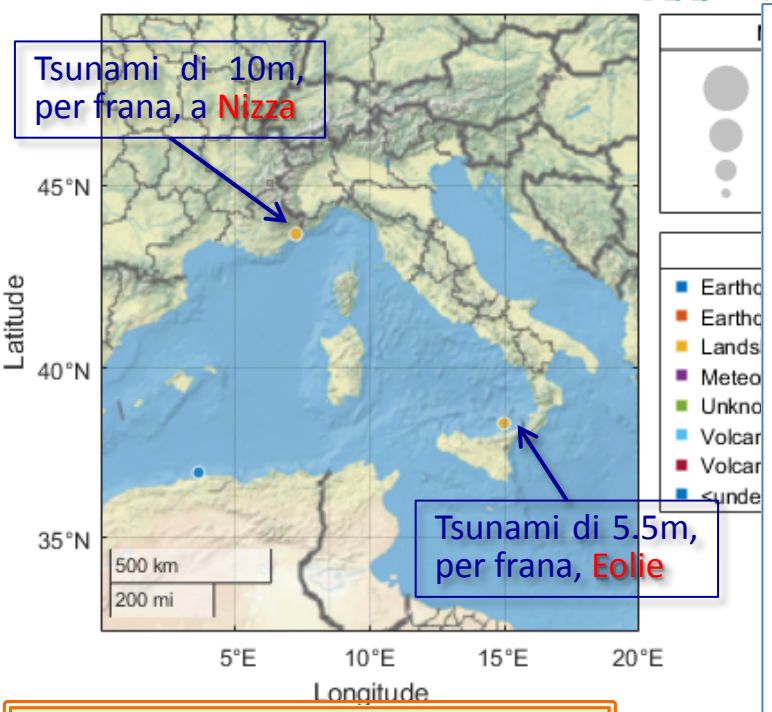

```
histogram(T{:,1})
```

produce errore perché i dati non sono numerici.



Creare mappe usando Latitudine e Longitudine

```
tsunamis=readtable('tsunamis.xlsx'); tsunamis.Properties.VariableNames
ans = 1×20 cell array
Columns 1 through 8
  {'Latitude'}  {'Longitude'}  {'Year'}  {'Month'}  {'Day'}  {'Hour'}  {'Minute'}  {'Second'}
Columns 9 through 14
  {'ValidityCode'}  {'Validity'}  {'CauseCode'}  {'Cause'}  {'EarthquakeMagni…'}  {'Country'}
Columns 15 through 20
  {'Location'}  {'MaxHeight'}  {'IidaMagnitude'}  {'Intensity'}  {'NumDeaths'}  {'DescDeaths'}
tsunamis.Cause=categorical(tsunamis.Cause);
geobubble(tsunamis,'Latitude','Longitude','SizeVariable','MaxHeight', ...
'ColorVariable','Cause')
geobasemap landcover
```




```
geolimits([35 47],[0 20])
```