Laboratorio di R - 1ª lezione

Prof. Mauro Gasparini

0. Preliminari

```
# in R il simbolo "#" indica l'inizio di una linea di commento
```

Uscire da R

```
q()
```

Chiedere aiuto ad R

```
help()                # chiede l'help generale
help(mean)            # chiede l'help del comando "mean"
?mean                 # equivalente a help(mean)
help.start()          # lancia l'help in un browser
```

Contenuto e manipolazione della cartella di lavoro

```
getwd()         # restituisce il nome della cartella di lavoro corrente
setwd(dir)      # pone "dir" come cartella di lavoro
```

Esempio:

```
> getwd()
[1] "C:/Programmi/R/R-2.2.0"
> dir <- "C:/directory_che_si_vuole"
> setwd(dir)
> getwd()
[1] "C:/ directory_che_si_vuole"

ls()                  # mostra il contenuto del workspace

# salva il workspace in un file chiamato "prova.rda"
save(list=ls(), file = "prova.rda")
# rda è l'estensione standard per indicare un file dati di R
# ma si può usare qualsiasi altra estensione o ometterla

rm(i)                 # elimina la variabile "i"
rm(list = ls())       # rimuovi tutto il contenuto del workspace
ls()

load("prova.rda")     # ricarichiamo i dati salvati in precedenza
ls()                  # ora nel workspace compaiono le variabili salvate
```

1. Aritmetica di base

Operatori aritmetici

In R, qualunque cosa venga scritta al prompt viene valutata:

```
> 1+2+3
[1] 6
> 2+3*4
[1] 14
> 3/2+1
[1] 2.5
> 2+(3*4)
[1] 14
> (2 + 3) * 4
[1] 20
```

```
> 4*3^3          # Usa ^ per calcolare un elevamento a potenza
[1] 108
```

R fornisce anche tutte le funzioni che si trovano su un calcolatore tascabile:

```
> sqrt(2)             # radice quadrata di 2
[1] 1.414214
> sin(pi)             # sin(pi greco) è zero
[1] 1.224606e-16      # e il risultato è vicino
```

Ecco una breve lista di funzioni:

Nome	**Operazione**
sqrt	radice quadrata
abs	valore assoluto
sin cos tan	funzioni trigonometriche
asin acos atan	funzioni trigonometriche inverse
exp log	esponenziale e logaritmo naturale

Le funzioni possono essere annidate:

```
> sqrt(sin(45*pi/180))
[1] 0.8408964
```

2. Vettori e matrici

Assegnazione di valori

Si può assegnare un valore ad una variabile utilizzando il comando **<-**

```
> x <- 4              # assegna a x il valore 4
> x                   # visualizza il contenuto della variabile x
[1] 4
```

Creazione di vettori:

La funzione c:

Con il comando `c(.. , .. , ..)` viene creato un vettore creato dagli elementi tra parentesi separati da virgole:

```
y <- c(2,7,4,1)       # assegna a "y" il vettore (2,7,4,1)
> y
[1] 2 7 4 1
```

Per costruire un vettore di stringhe, basta concatenare stringhe di caratteri delimitati da virgolette:

```
y <- c("Questo", "e'", "un esempio")
y
[1] "Questo"     "e'"         "un esempio"
```

È possibile creare vettori logici:

```
x <- c(5<2, 3>1, 1==0)
x
[1] FALSE  TRUE FALSE
```

La funzione `scan`:

Se si devono immettere tanti dati, conviene utilizzare la funzione `scan`:

```
> x <- scan()
1: 1
2: 3
3: 6
4: 7
5: 99
```

```
6: 43
7:
Read 6 items
> x
[1]  1  3  6  7 99 43
```

`scan()` può anche servire per leggere un vettore da un file:

```
prova <- scan("data.dat")
```

Le funzioni `seq` e `rep`:
`seq (min, max, incremento)` viene utilizzato per creare delle successioni di numeri che variano da `min` a `max` con un passo dato da `incremento`:

```
> seq(-3,6,2)         # successione di valori da -3 a 6, passo 2
[1] -3 -1  1  3  5

> seq(-3,-1,length=11) # successione di valori da -3 a -1 di lunghezza 11
[1] -3.0 -2.8 -2.6 -2.4 -2.2 -2.0 -1.8 -1.6 -1.4 -1.2 -1.0

> 1:10
[1]  1  2  3  4  5  6  7  8  9 10
> seq(1,10,1)                        # le due scritture sono equivalenti
[1]  1  2  3  4  5  6  7  8  9 10
```

La funzione `rep(schema, n)` viene usata per ripetere un determinato schema `n` volte:

```
> rep(1,5)
[1] 1 1 1 1 1
> rep(c(0,5), 3)
[1] 0 5 0 5 0 5
```

Operazioni che coinvolgono vettori:
Per i vettori vale la stessa aritmetica di base degli scalari:

```
> x <- c(1,2,3,4)
> x
[1] 1 2 3 4
> y <- c(2,4,6,8)
> y
[1] 2 4 6 8
> x + y               # operazioni termine a termine
[1]  3  6  9 12
> x – y
[1] -1 -2 -3 -4
> x * y
[1]  2  8 18 32
> x / y
[1] 0.5 0.5 0.5 0.5
# R è un linguaggio case-sensitive: nel nostro caso esiste "y" ma non "Y"
> x * Y               # dà messaggio di errore
Errore: oggetto "Y" non trovato

# "t()" indica la funzione "trasposto di"
# Il simbolo "%*%" è l'operatore prodotto matriciale

> t(y)%*% y           # vettore riga * vettore colonna = scalare
     [,1]
```

```
[1,]  120
```

Alcune funzioni utili per la manipolazione di vettori:
Dato un vettore x:

		Esempio
`x`	vettore	`seq(3,26,1)`
`length(x)`	numero di elementi di x	`24`
`max(x)`	massimo valore del vettore x	`26`
`min(x)`	minimo valore del vettore x	`3`
`sum(x)`	somma dei valori in x	`348`
`prod(x)`	prodotto dei valori in x	`2.016457e+26`
`mean(x)`	media aritmetica	`14.5`
`median(x)`	mediana aritmetica	`14.5`
`var(x)`	varianza campionaria	`50`

Esercizio 1:
Costruire un vettore **x** con elementi: 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 8.5, 9, 9.5 e 10.
Per **x** calcolare: lunghezza, somma degli elementi, prodotto degli elementi, media e varianza.

Matrici e operazioni algebriche
Le matrici vengono create in R tramite la funzione `matrix(data, nrow, ncol, byrow = F)`; notare che di default R costruisce le matrici per COLONNE: tale comportamento si può invertire ponendo `byrow = T`.

```
> A <- matrix(data = 1:30, nrow = 5, ncol = 6, byrow = FALSE)
> A
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    6   11   16   21   26
[2,]    2    7   12   17   22   27
[3,]    3    8   13   18   23   28
[4,]    4    9   14   19   24   29
[5,]    5   10   15   20   25   30

> matrix(0,2,3)       # matrice di 0
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
> matrix( ,2,3)       # riempimento senza assegnazione di valori (NA)
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA

> x <- matrix(c(1,2,3,4),1,4)   # "x" definito come vettore riga
> x
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
> y <- matrix(c(1,2,3,4),4,1)   # "y" definito come vettore colonna
> y
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4

> diag(1,3,3)         # crea una matrice diagonale di 1 e dimensioni 3x3
```

```
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
> matrix("A",2,3)          # crea una matrice 2x3 formata da simboli "A"
     [,1] [,2] [,3]
[1,] "A"  "A"  "A"
[2,] "A"  "A"  "A"
```

Esercizio 2:
Dai vettori:

```
x <- c(3, 5, 9, 1, 2, 10, 12, 24, 6)
y <- c(0.15, 0, 0.32, 0.51, 0.18, 0.22, 0.6, 0.98, 0.12)
z <- c(123, 415, 981, 643, 1080, 89, 46, 75, 910)
```

si costruisca la matrice che ha come colonne i tre vettori.
Si calcoli poi la media degli elementi di **z** che sono minori di 900.

Accedere agli elementi dei vettori e delle matrici
Gli elementi di vettori e di matrici si possono estrarre utilizzando le parentesi quadre []

```
> x <- 1:10
> x[3]                     # terzo elemento del vettore x
[1] 3
> x[2:5]                   # si possono estrarre sottoinsiemi di elementi
[1] 2 3 4 5
> x[-4]                    # si omette il quarto elemento del vettore x
[1]  1  2  3  5  6  7  8  9 10

> A <- matrix(1:30, 5, 6)
> A
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    6   11   16   21   26
[2,]    2    7   12   17   22   27
[3,]    3    8   13   18   23   28
[4,]    4    9   14   19   24   29
[5,]    5   10   15   20   25   30
> A[2,3]              # elemento (2,3) della matrice A
[1] 12
> A[2, ]              # seconda riga della matrice A
[1]  2  7 12 17 22 27
> A[ ,6]              # sesta colonna della matrice A
[1] 26 27 28 29 30
> A[c(1,4), c(2,3)]  # righe 1 e 4, colonne 2 e 3 della matrice A
     [,1] [,2]
[1,]    6   11
[2,]    9   14
```

Ricerca di elementi all'interno di un vettore: il comando `which`

```
> x <- -3:8
> x
[1] -3 -2 -1  0  1  2  3  4  5  6  7  8
> which(x < 2)  #fornisce le posizioni degli elementi che verificano x<2
[1] 1 2 3 4 5
# posizioni degli elementi che verificano x >= -1 e x < 5
> which((x >= -1) & (x < 5))
[1] 3 4 5 6 7 8
# posizioni degli elementi che verificano x < -2  o x > 1
```

```
> which((x < -2) | (x > 1))
[1]  1  6  7  8  9 10 11 12
```

Esercizio 3:
Dal vettore:
`x <- c(20, 3, 26, 14, 22, 14, 5, 24, 28, 21, 17, 29, 24, 1, 25, 26, 27, 7, 11, 25, 20, 25, 9, 16, 20, 7, 19, 27, 30, 2)`
si crei una matrice `mat` di 3 colonne, procedendo per riga.
Si costruisca poi una matrice `mat1` selezionando le prime 8 righe di `mat` e per `mat1` si calcolino media, varianza e mediana degli elementi della prima colonna ai quali corrispondono elementi della seconda colonna maggiori di 10.

---

3. Dataframe

Un dataframe (chiamato a volte matrice dei dati) è un oggetto simile ad una matrice, ma usato per rappresentare dati sperimentali.
Ogni riga rappresenta un'unità statistica, ogni colonna rappresenta una variabile misurata sulle unità statistiche.
Le colonne possono contenere variabili numeriche o categoriali.
Per leggere un insieme di dati di questo tipo si usa la funzione `read.table()`, che automaticamente controlla se le variabili sono numeriche o qualitative, se le righe e/o le colonne hanno etichette.

Esempio:
Si supponga che il file `exams.txt` sia così costituito:

```
       name          age    mark1  mark2  mark3
1      arthur        21     78     77     56
2      barry         22     85     65     81
3      charlie       19     56     66     85
4      dave          22     75     75     54
```

Possiamo acquisirlo con il comando:

```
> students <- read.table("exams.txt")
> students
```

R nota che la prima riga ha un elemento in meno delle altre; quindi interpreta la prima riga come una riga di etichette per le colonne.

Si possono cambiare le etichette con il comando:

```
> names(students) <- c('name','age','test1','test2','test3')
```

Accesso alle colonne di un dataframe

```
> students$name
[1] arthur  barry   charlie dave
Levels: arthur barry charlie dave
> students [1, ]
    name age mark1 mark2 mark3
1 arthur  21    78    77    56
> students [ ,3]
[1] 78 85 56 75
> students [1,3]
[1] 78

# "attach" rende disponibili direttamente le colonne di un dataframe
```

```
> attach(students)
> name
[1] arthur  barry   charlie dave
Levels: arthur barry charlie dave
> age
[1] 21 22 19 22
```

Per avere delle statistiche di base sulle variabili contenute in `students` si usa la funzione:

```
> summary(students)
      name        age           mark1           mark2           mark3      
 arthur :1   Min.   :19.0   Min.   :56.00   Min.   :65.00   Min.   :54.0  
 barry  :1   1st Qu.:20.5   1st Qu.:70.25   1st Qu.:65.75   1st Qu.:55.5  
 charlie:1   Median :21.5   Median :76.50   Median :70.50   Median :68.5  
 dave   :1   Mean   :21.0   Mean   :73.50   Mean   :70.75   Mean   :69.0  
             3rd Qu.:22.0   3rd Qu.:79.75   3rd Qu.:75.50   3rd Qu.:82.0  
             Max.   :22.0   Max.   :85.00   Max.   :77.00   Max.   :85.0  
```

Estrazione di elementi da un dataframe

Per estrarre elementi da un dataframe valgono le stesse regole delle matrici:

```
> mark1
[1] 78 85 56 75 66

# Estrazione degli studenti con un punteggio > 70 nel test 1
> students[mark1 > 70, ]
    name age mark1 mark2 mark3 mark4
1 arthur  21    78    77    56    58
2  barry  22    85    65    81    76
4   dave  22    75    75    54    65
```

Esercizio 4:

Si costruisca il dataframe:

```
> x
    a    b
1   2    I
2   3   II
3   4  III
```

A partire dal dataframe creato si acceda alla seconda colonna di `x`.

---

4. Alcuni strumenti grafici

Un valido strumento per l'analisi preliminare di dati quantitativi è la rappresentazione grafica tramite diagrammi statistici.

Per aprire una nuova finestra grafica si usa l'istruzione:

```
> win.graph()
```

Una finestra grafica può essere suddivisa in varie figure usando il comando:

```
par(mfrow = c(n,m))
```

con cui le figure vengono disposte su `n` righe e `m` colonne sulla finestra grafica attiva.

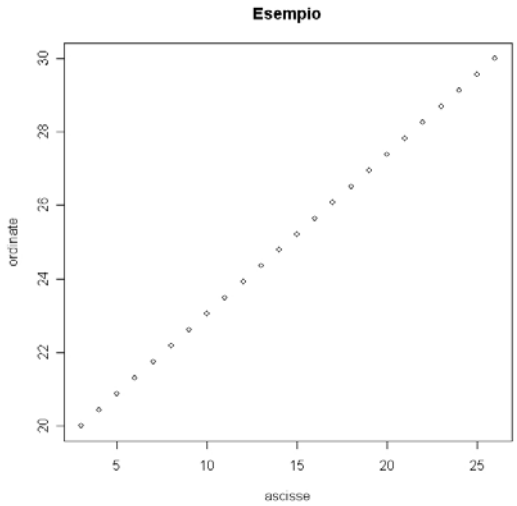
Diagramma di dispersione:

Il diagramma di dispersione di **y** (vettore ordinate) rispetto a **x** (vettore ascisse) si ottiene con `plot(x, y)`.

```
?plot           # guarda la varie opzioni del comando plot
```

Esempio:

```
> x <- seq(3,26,1)
> y <- seq(20,30,length = 24)
> plot(x, y, xlab = "ascisse", ylab =
"ordinate", main = "Esempio")
```



Istogramma:
Per creare un istogramma dei dati contenuti nel vettore `x` si usa il comando `hist`:

```
hist(x)                    # istogramma con frequenze assolute
hist(x, freq = FALSE)      # istogramma con frequenze relative
```

Per creare una **tabelle delle frequenze** dei dati contenuti nel vettore `x`:
`table(x)`

Box plot:
Per creare un boxplot dei dati contenuti nel vettore `x` si usa il comando:
`boxplot(x)`

Si può accedere ad alcune statistiche del boxplot in questo modo:

```
> a <- boxplot(x)
> a
$stats
      [,1]
[1,] (min valore no outlier)
[2,] (1° quartile)
[3,] (2° quartile - mediana)
[4,] (3° quartile)
[5,] (max valore no outlier)

$n
[1] (n° osservazioni)

$out
[1] (eventuali outlier)
```

Esercizio 5:
Importa in R i dati del file di testo USA`.txt` (es. 2.7 del Ross, svolto a lezione).
Rappresenta in un istogramma (frequenze relative) il numero di vittime all'anno.

Esercizio 6:
Importa in R i dati del file di testo `cani.txt` (es. 2.23 del Ross, svolto a lezione).
Rappresenta i dati dei cani iscritti in un box plot, verificando che i valori dei quartili ottenuti in R coincidano con quelli ottenuti durante l'esercitazione.
