# INTRODUZIONE A R
## Lezione 1

Silvia Bacci*e Silvia Pandolfi†

## 1 Introduzione

R è un ambiente di elaborazione che consente di manipolare dati, eseguire calcoli e procedure complesse nonché rappresentare graficamente i risultati.

R è un linguaggio di programmazione *object oriented*, ovvero tutte le operazioni vengono svolte su e tramite oggetti e a loro volta ne generano.

R è un software completamente gratuito, scaricabile dal sito internet http://www.r-project.org/. Insieme al software si possono scaricare dal sito principale e dai numerosi siti collegati sia manuali d'uso che numerosi pacchetti aggiuntivi.

I moduli di R sono organizzati in un apposito sito detto CRAN (http://cran.r-project.org/) (CRAN = Comprehensive R Archive Network), in analogia al CTAN [è l'acronimo di Comprehensive TEX Archive Network] e CPAN [l'acronimo di Comprehensive Perl Archive Network]).

R è un linguaggio di programmazione derivato dal linguaggio S e può implementare in sè moduli creati da altri linguaggi come il C , il C++, il Fortran.

All'avvio di R appare semplicemente una console con un cursore, rappresentato dal simbolo ">" (linea di comando) dove inserire i comandi e le funzioni.

Oltre al cursore è presente il *Text Editor* dove copiare i comandi digitati (*File* → *New Script*). Per salvare il file comandi basta utilizzare l'apposito menù a tendina (estensione .R).

Il *workspace* è un ambiente virtuale che contiene tutti i risultati delle elaborazioni, i dati, le variabili e tutti gli oggetti creati da tastiera. R produce due tipi di dati:

- `.R` : è l'equivalente di un file testo, creato attraverso il text editor di R, dove inserire i comandi da lanciare e le funzioni
- `.Rdata`: raccoglie gli oggetti (ovvero le variabili, le matrici, i vettori che sono stati creati). È l'equivalente del file `.mat` in MATLAB

R è *case sensitive*, cioè considera caratteri differenti lettere maiuscole e minuscole.

*silvia.bacci@unipg.it
†pandolfi@stat.unipg.it

# 2 Primi passi

La console di R può essere utilizzata come semplice calcolatrice. È un ambiente interattivo, ossia i comandi producono una risposta immediata:

```
> #somma
>2+2
[1] 4

> #moltiplicazione
 >2*2
[1] 4

> #sottrazione e divisione
> (2 - 3)/6
[1] -0.167

 > #elevamento a potenza
> 2^2
[1] 4
```

Esistono diverse funzioni per le principali operazioni matematiche e trigonometriche. Per dare solo qualche esempio, `sqrt` serve per estrarre la radice quadrata da un numero, `log` calcola il logaritmo naturale ed `exp` l'esponenziale:

```
> sqrt(9)
[1] 3

> log(1)
[1] 0

> exp(0)
[1] 1
```

# 3 Gli oggetti in R

Ogni entità che il programma crea e manipola è definita un **oggetto**, che può essere un numero, una variabile, una funzione, o più in generale, strutture costruite a partire da tali componenti.

I diversi tipi di variabili che R può gestire si distinguono in: **numeri reali**, **numeri complessi**, **stringe di testo**, e **valori logici**. I numeri complessi si distinguono per la presenza della parte immaginaria, che deve sempre essere esplicitata, ad esempio:

```
> sqrt(-17+0i)
```

Le stringhe, cioè insiemi di caratteri, sono racchiuse da doppie virgolette:

```
> "testo"
```

I valori logici disponibili sono `TRUE`, `FALSE` e `NA` che indica "risposta non disponibile". I primi due valori possono essere abbreviati rispettivamente con `T` e `F` (lettere maiuscole!).

Si possono assegnare oggetti in R indifferentemente con "=" o con "$<-$".

```
> prova = 19
> prova
> prova <- 19
> 19 -> prova
> Prova = 7
> Prova
> PROVA = 10
> PROVA
> prova1 <- "pippo"
> prova1
> prova2 <- 3>5
> prova2
```

I nomi degli oggetti non devono contenere spazi vuoti, simboli matematici e non devono iniziare con un numero:

```
> pro va = 2
Error: unexpected symbol in "pro va"
> x-y = 2
Error in x - y = 2 : could not find function "-<-"
> 2x = 2
Error: unexpected symbol in "2x"
> prova\% <- 6
Error: unexpected input in "prova\% <- 6"
```

Alcuni nomi sono "protetti" da R:

```
> for = 3
Error: unexpected "=" in "for ="
```

Tutte le lettere ed i numeri possono essere utilizzati per dare nomi a variabili, funzioni e procedure; sono però esclusi gli spazi e tutti i segni di punteggiatura ad eccezione del punto (“.”), comunemente usato per separare le parole che compongono il nome; ogni nome deve cominciare con una lettera e non con un numero.

Ogni comando deve essere separato da un punto e virgola (“;”) oppure deve essere su una nuova linea.

Testi di commento possono essere aggiunti dopo il simbolo cancelletto (“#”), tutto ciò che si trova dopo questo simbolo e fino alla riga successiva viene ignorato da R.

Se si va a capo prima della conclusione di un comando R mostrerà all’inizio della riga il simbolo più (“+”).

È possibile richiamare i comandi impartiti in precedenza utilizzando la freccia “sù” della tastiera.

# 4 Primi comandi

I primi comandi da imparare sono:

```
>#  Fornisce la lista dei nomi degli oggetti in memoria
>ls()
[1] "prova"  "Prova"  "PROVA"  "prova1" "prova2"

>#Rimuove tutti gli oggetti tra parentesi
> rm(prova, PROVA)
> ls()
[1] "Prova"  "prova1" "prova2"

> #Cancella tutti gli oggetti nello spazio di lavoro
> rm(list=ls(all=TRUE)) # oppure: rm(list=ls())
> ls()
character(0)
```

Ogni installazione di R contiene ottima ed abbondante documentazione. Inoltre, è disponibile un help in linea consultabile con la funzione `help()`, mentre `help.search()` consente di effettuare ricerche per parole chiavi nella documentazione in linea.

```
> #Richiama in una nuova finestra le spiegazioni associate alla funzione
> #o al comando specificati
> help(nome.funzione)

> #oppure
> ?nome.funzione

> #Esempio:
> help(cbind)
> ?cbind

> #Cerca la stringa "parola chiave" tra la documentazione disponibile
> help.search("parola chiave")

> #Richiama l'help in formato html
> help.start()
```

Le funzioni di R, come si può osservare dall'help, si compongono in:

```
funzione(argomento1 = ..., argomento2 = ..., ...)
```

L'ordine degli argomenti è importante all'interno di una funzione se non viene indicato il nome dell'argomento, mentre se questo viene specificato, la sua posizione è indifferente.

# 5 Lo spazio di lavoro

Gli oggetti creati dall'utente vengono temporaneamente salvati nello spazio di lavoro (*workspace*).

- `getwd()` : mostra il *path* della cartella nel computer che è stata automaticamente selezionata al momento dell'installazione del programma.
- `setwd()`: per modificare lo spazio di lavoro, e cambiare directory. In alternativa, dal menu "File", si sceglie l'opzione "Cambia directory" e si sceglie la la cartella.
- `save.image()`: per salvare tutto lo spazio di lavoro in maniera definitiva. Crea un file senza nome nella cartella selezionata con estensione `.Rdata`.
  Con `save.image("nomefile.Rdata")` si salva lo spazio di lavoro con un nome specifico. Ad esempio:

  ```
  save.image("output_prova.RData")
  ```

- `save()`: salva solo alcuni oggetti nello spazio di lavoro. Ad esempio:
  - `save(x, file = "x.Rdata")`
  - `save(x, y, z, file = "prova.Rdata")`
- `load(".Rdata")` o `load("prova.Rdata")`: importa il file `.Rdata` in una nuova sessione di lavoro.

  ```
  rm(list = ls())
  ls()
  load("output_prova.RData")
  ls()
  ```

- `dir()`: mostra il nome dei file presenti nella directory di lavoro.
- `source("comandi.R")`: esegue tutti i comandi contenuti nel file "comandi.R".
- `history(50)`: mostra gli ultimi 50 (il default è 25) comandi digitati alla console.
- `savehistory`: salva tutti i comandi della sessione di lavoro sul file `.Rhistory` (default).

# 6 Tipologie di dati in R

R lavora con dati strutturati. Le strutture di dati contemplate sono:

- *variabili*: contengono un singolo dato
- *vettori*: insieme lineare di elementi omogenei per tipologia (tutti numeri, tutte stringhe, ecc.)
- *fattori*: vettore utilizzato per classificare o suddividere in livelli gli elementi di un altro vettore
- *array e matrici*: sono vettori multidimensionali, le matrici hanno due dimensioni: righe e colonne. Gli array sono insiemi di numeri con $p$ dimensioni
- *data frame*: sono matrici bidimensionali dove ogni colonna può avere un tipo di dato diverso dalle altre
- *liste*: insieme di elementi non omogenei tra loro

## 6.1 Vettori di dati

Per definire un vettore esistono molti modi, il più comune dei quali è la funzione `c()` che combina i valori presenti nel suo elenco in un vettore.

```
> #†definizione di un vettore (colonna)
> x = c(1,2,3,4,5,6,7,8,9,10)
> x
[1] 1 2 3 4 5 6 7 8 9 10
> t(x) # trasposto (vettore riga)

> z = c("pippo", "topolino", "pluto", "paperino")
> z
[1] "pippo" "topolino" "pluto" "paperino"

> w = c(x, z) # concateno due vettori
> w
  [1] "1"        "2"        "3"        "4"        "5"        "6"
  [7] "7"        "8"        "9"        "10"       "pippo"    "topolino"
 [13] "pluto"    "paperino"
```

Per selezionare gli elementi di un vettore si deve inserire tra parentesi quadre, successivamente al nome del vettore, un indice, o un vettore di indici, che indichi la posizione degli elementi del vettore da estrarre. Se davanti all'indice c'è il segno "-", il corrispondente elemento viene eliminato dal vettore:

```
> x[1]
[1] 1

> x[c(6, 3)]
```

```
[1] 6 3

> z[2:4]
[1] "topolino" "pluto"    "paperino"

> x[- 4]
[1]  1  2  3  5  6  7  8  9 10
```

### 6.1.1 Aritmetica con i vettori

I vettori possono essere utilizzati in qualunque espressione aritmetica. In questo caso l'operazione è applicata a tutti gli elementi del vettore. Ad essi possono essere applicate anche funzioni statistiche:

```
> campione <- c (6, 1, 5, 9, 4, 7, 8, 2, 5, 8)

> campione^2 #elevamento al quadrato
[1] 36  1 25 81 16 49 64  4 25 64

> min(campione)  #valore minimo
[1] 1

> max(campione)  #valore massimo
[1] 9

> length(campione)  #numero di valori (lunghezza del vettore)
[1] 10

> sum(campione) #somma degli elementi
[1] 55
```

Provare inoltre

```
mean(campione) #media
median(campione) #mediana
range(campione) #restituisce un vettore di due elementi, il min e max
sd(campione) #deviazione standard
var(campione) #varianza
sum((campione - mean(campione))^2)/(length(campione)-1)

?var #apre l'help della funzione var
```

Si noti che per le funzioni statistiche, così come per tutte le funzioni, i dati e gli eventuali parametri sono racchiusi tra parentesi e che la funzione di per sé restituisce un valore, o un insieme di valori. Negli esempi sopra, il risultato dell'operazione viene restituito, cioè visualizzato a schermo, ma non memorizzato in alcuna variabile (non viene assegnato). Gli ultimi due valori ottenuti sono identici: l'ultima operazione infatti è lo stesso calcolo della varianza eseguito dalla funzione `var()`.

Altre operazioni utili con i vettori sono:

```
> sort(campione) #stessi valori del vettore ma in ordine crescente
 [1] 1 2 4 5 5 6 7 8 8 9

> order(campione) #restituisce l'ordine dei valori nel vettore
 [1]  2  8  5  3  9  1  6  7 10  4
```

Quando si compiono operazioni tra due vettori, il primo elemento dell'uno viene operato con il primo elemento dell'altro, il secondo col secondo e così fino alla fine. Se un vettore è più corto dell'altro, quando termina viene riutilizzato dall'inizio.

```
> a = c (10,10,10,10,10,10,10,10,10,10)
> b = c (1,2)
> a*b
[1] 10 20 10 20 10 20 10 20 10 20
```

### 6.1.2 Tipologie particolari di vettori: le funzioni `seq` e `rep`

- `seq(from = 1, to, by, length)` : genera un vettore che crea una sequenza regolare di numeri da `from` fino a `to`; `by` indica il passo della sequenza, mentre `length` la lunghezza.
- `rep(x, times)`: genera un vettore in cui un oggetto (`x`), numero o vettore, viene ripetuto un numero di volte pari a `times`

```
> #due diversi modi di ottenere la stessa sequenza
> x = seq(from = 3, to = 15, by = 3)
> x1 = seq(from = 3, to = 15, length = 5)

> seq(2, 10, 2)
[1] 2 4 6 8 10

> #se gli argomenti sono specificati possono anche essere posti in ordine diverso
> seq(1, length = 5, by = .5)
[1] 1.0 1.5 2.0 2.5 3.0

> #modo compatto per ottenere delle sequenze di passo 1
> 1:5
[1] 1 2 3 4 5

> 5:1
[1] 5 4 3 2 1

> rep(x = 1, times = 5)
[1] 1 1 1 1 1

> rep(c(1, 2), 2)
[1] 1 2 1 2
```

```
> c(rep(1,3),4,5,rep(2,4))
[1] 1 1 1 4 5 2 2 2 2
```

### 6.1.3 Operatori logici

I valori logici sono normalmente il risultato di operazioni logiche. Gli operatori logici utilizzabili in R sono:

```
<    #minore
<=   #minore o uguale
>    #maggiore
>=   #maggiore o uguale
==   #uguale
!=   #diverso
&    #è l'intersezione ("e")
|    #è l'unione ("o")
!    #è la negazione ("non")
```

In R i valori mancanti sono rappresentati dal valore `NA` ("Not Available"). Le operazioni logiche eseguite su valori mancanti o comunque non utilizzabili, restituiscono ancora il valore `NA`.

Operazioni aritmetiche "impossibili", come ad esempio le divisioni per zero, restituiscono il valore `NaN` che significa che "non è un numero" ( "Not a Number").

`Inf` e `-Inf` rappresentano l'infinito positivo e negativo (insieme a `NA` e `NaN`, sono parole riservate di R.

Alcuni esempi:

```
> 2 * 2 == 4
[1] TRUE

>2*2>4
[1] FALSE

> 2 * 2 >= 4
[1] TRUE

> 2 * 2 != 4
[1] FALSE

> a=1:10
> a <= 5
 [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE

> a[a>5]
[1]  6  7  8  9 10

> a[a>=3 & a<7]
```

```
[1] 3 4 5 6

> a[a<3 | a>8]
[1]  1  2  9 10

> a[a != 10]
[1] 1 2 3 4 5 6 7 8 9

> 0 / 0
[1] NaN

> 1/0
[1] Inf

> x <- c(1,2,NA,3)
> mean(x)
[1] NA
```

Per cercare all'interno di un vettore un determinato elemento si può applicare il comando `which()`, il cui output è l'indice di posizione. Mettiamo di voler conoscere la posizione di pippo in z:

```
> z = c("pippo", "pluto", "paperino")
> which(z == "pippo")
[1] 1
```

Come si può osservare, il comando `which()` richiede la specificazione di una condizione logica, in questo caso quella di uguaglianza. Mettiamo ora di voler individuare tutti gli elementi tranne pluto:

```
> which(z != "pluto")
[1] 1 3

> a = -10:5
> a
 [1] -10  -9  -8  -7  -6  -5  -4  -3  -2  -1   0   1   2   3   4   5
> which(a>0)
[1] 12 13 14 15 16
```

Infine vediamo come imporre una doppia condizione al comando `which`. Nel caso in cui si voglia porre che entrambe le condizioni siano rispettate disgiuntamente (operazione "unione"), il separatore logico è | (oppure):

```
> which(a< -5 | a>2)
[1]  1  2  3  4  5 14 15 16
```

### 6.1.4 Esercizio

Creare i seguenti vettori:

```
ETA = c(19, 22, 21, 23, 22, 20)
PESO = c(50, 75, 80, 56, 75, 58)
ALTEZZA = c(1.65, 1.78, 1.91, 1.72, 1.81, 1.68)
SESSO = c("F", "M", "M", "F", "M", "F")
PROV = c("RM", "RM", "TO", "NA", "TO", "NA")
```

- Calcolare l'altezza in centimetri
- Calcolare il body mass index (BMI), dato dal rapporto tra il peso e il quadrato dell'altezza
- Arrotondare BMI a due cifre decimali con la funzione `round()` (cercare nell'help)
- Selezionare il peso della prima unità
- Eliminare l'altezza della seconda e terza unità
- Estrarre l'altezza di chi ha un peso minore o uguale a 70 Kg
- Estrarre il peso di chi ha un'altezza maggiore di 1.70
- Estrarre l'altezza di chi ha 22 anni
- Estrarre il peso delle unità di sesso femminile
- Estrarre l'altezza delle unità che hanno meno di 22 anni e che non sono nate nella provincia di Napoli

## 6.2 Fattori

I fattori sono vettori utilizzati per classificare o suddividere in livelli gli elementi di un altro vettore di pari lunghezza. Per trasformare un vettore, contenente le etichette dei livelli, in "fattore" si usa la funzione `factor()`.

```
> SESSO = factor(SESSO)
> SESSO
[1] F M M F M F
Levels: F M

> PROV = factor(PROV)
> PROV
[1] RM RM TO NA TO NA
Levels: NA RM TO

> #per ottenere l'elenco dei livelli presenti nel fattore
> levels(PROV)

> is.factor(SESSO)
[1] TRUE

> #trasforma le variabili di classificazione in numeriche
> as.integer(PROV)
```

## 6.3 Array e Matrici

Gli array - `array()` - sono vettori multidimensionali. Le matrici - `matrix()` - sono array bidimensionali (righe e colonne).

```
> dati = c(1,4,6,32,6,7,4,6,8,5,3,6,7,67,4)
> righe = 5
> colonne = 3

> matrice  = matrix(data=dati,nrow=righe,ncol=colonne)
> matrice
     [,1] [,2] [,3]
[1,]    1    7    3
[2,]    4    4    6
[3,]    6    6    7
[4,]   32    8   67
[5,]    6    5    4

> matrice2  = matrix(data=dati,nrow=righe,ncol=colonne,byrow=TRUE)
> matrice2
     [,1] [,2] [,3]
[1,]    1    4    6
[2,]   32    6    7
[3,]    4    6    8
[4,]    5    3    6
[5,]    7   67    4

> matrice3 <- array(data=dati,dim=c(righe,colonne))
> #restituisce la dimensione dell'array o matrice
> dim(matrice3)
[1] 5 3

> m4 = matrix(1:15,ncol=3)
> m5 = matrix(1,3,4)
> m6 = array(m4, c(3,3,2))
> dim(m6)
[1] 3 3 2

> #seleziona un sottoinsieme di elementi di un array o matrice
> matrice[1,2] # estrazione dell'elemento sulla riga 1 e la colonna 2
> matrice[4,3]
> matrice[5,] # estrazione della riga 5
> matrice[,2] # estrazione della colonna 2
> matrice[4:5,2]
```

## 6.4 Data frame

Sono matrici in cui le colonne possono essere di natura diversa. Nelle rilevazioni statistiche, le colonne sono le caratteristiche rilevate (variabili), mentre le righe costituiscono le unità.

```
> data = data.frame(ETA,PESO,ALTEZZA,SESSO,PROV)

> # analizza la struttura di un oggetto
> str(data)

> # restituisce il nome delle variabili del data frame
> # che possono essere richiamate con il simbolo $
> names(data)
> data$ETA
> data[,1]
> data[-1,]  # elimina la prima riga

> iris    # data frame presente in R
> str(iris)

> # crea una variabile per ogni vettore (colonna) del dataframe
> attach(iris)
> Species
> detach(iris)
> Species
```

## 6.5 Liste

Sono insiemi di oggetti di natura e dimensione diversa. Sono spesso usate per raccogliere i risultati di un'analisi. Per generare una lista si impiega la funzione `list()`.

```
> x = c(1,3,6,15)
> lista = list(data,x)

> lista
[[1]]
  ETA PESO ALTEZZA SESSO PROV
1  19   50    1.65     F   RM
2  22   75    1.78     M   RM
3  21   80    1.91     M   TO
4  23   56    1.72     F   NA
5  22   75    1.81     M   TO
6  20   58    1.68     F   NA

[[2]]
[1]  1  3  6 15

> #Estrae un oggetto da una lista
> lista[[1]]

> names(lista) = c("Dati","sequenza")
> lista
```

```
$Dati
  ETA PESO ALTEZZA SESSO PROV
1  19   50    1.65     F   RM
2  22   75    1.78     M   RM
3  21   80    1.91     M   TO
4  23   56    1.72     F   NA
5  22   75    1.81     M   TO
6  20   58    1.68     F   NA

$sequenza
[1]  1  3  6 15

> lista$sequenza
[1]  1  3  6 15
> lista[[2]]
[1]  1  3  6 15

> lista2=list(marito="Fred",moglie="Mary",figli=3,anni.figli=c(2,3,6))
> lista2
$marito
[1] "Fred"

$moglie
[1] "Mary"

$figli
[1] 3

$anni.figli
[1] 2 3 6

>#Restituisce la classe di un oggetto
> class(x)
[1] "numeric"

> class(lista2)
[1] "list"
```

## 6.6 Operazioni con vettori e matrici

Sui vettori è possibile effettuare operazioni matematiche

```
> b  = c(1,3,5,2)

> b[1] = 100  # sostituisco il valore assunto dal primo elemento di un vettore
> b
[1]100 3 5 2
```

```
> b-1
> b*2
> log(b)

> c = c(2,3,1,2)
> b*c   # prodotto elemento per elemento
[1]200 9 5 4

>#  prodotto scalare b'c (inner product): moltiplico ogni elemento di b
>#per il corrispondente di c e sommo; ottengo uno scalare
> t(b) %*% c
      [,1]
[1,]   218

> # bc' (outer product): moltiplico ogni elemento di b
>#  per tutti gli elementi di t(c); ottengo una matrice 4x4
> X = b %*% t(c)
     [,1] [,2] [,3] [,4]
[1,]  200  300  100  200
[2,]    6    9    3    6
[3,]   10   15    5   10
[4,]    4    6    2    4

> # genera una matrice 4x2 selezionando le colonne 1 e 4 di X
> # e concatenandole per colonna
>  Y = cbind(X[,1],X[,4])
> Y
     [,1] [,2]
[1,]  200  200
[2,]    6    6
[3,]   10   10
[4,]    4    4


> Y - cbind(X[,2],X[,3])
> Y * cbind(X[,2],X[,3])

> # l'analogo comando rbind() può essere utilizzato per concatenare le righe
> rbind(X[,1],X[,4])
     [,1] [,2] [,3] [,4]
[1,]  200    6   10    4
[2,]  200    6   10    4
```

Le precedenti operazioni sono state applicate a matrici della stessa dimensione, in quanto si tratta di operazioni elemento per elemento. Il prodotto matriciale, invece, richiede che il numero delle colonne della prima matrice sia uguale al numero delle righe della seconda matrice.

```
> dim(Y)
```

```
[1] 4 2
> t(Y)    #la funzione t() traspone Y
     [,1] [,2] [,3] [,4]
[1,]  200    6   10    4
[2,]  200    6   10    4
> t(Y)%*%cbind(X[,2],X[,3])    # prodotto matriciale
      [,1]  [,2]
[1,] 60228 20076
[2,] 60228 20076
```

In R esiste una vasta libreria di funzioni di Algebra Lineare che si applicano a matrici o a vettori. Le più diffuse sono:

- `diag()`: estrae o sostituisce gli elementi diagonali di una matrice, oppure costruisce una matrice diagonale
- `det()`: calcola il determinante di una matrice
- `solve()`: risolve un sistema lineare di equazioni (inverte anche una matrice)

```
> diag(X)
> diag(4)
> diag(rep(2,3))
> Z = matrix(c(3,4,6,7,1,12,4,9,21),3,3)
> det(Z)
> Zinv = solve(Z)
```