Laboratorio di ST1 - Lezione 1

Antonietta di Salvatore

Dipartimento di Matematica
Universitá degli Studi Roma Tre

Outline

In questa prima esercitazione si faranno i primi passi con **R**.
In particolare ci focalizzeremo su

- l'istallazione
- il linguaggio
- gli oggetti
- prime semplici statistiche
- help
- memoria e salvataggio

Perché R

- Il programma é distribuito gratuitamente attraverso internet sotto la forma di GNU General Public License (programma open source)
- La grandezza della sua comunitá scientifica
- Si sta diffondendo anche in ambito aziendale come strumento operativo
- Il programma gira sotto i principali sistemi operativin(Windows, Mac OS X, Unix e nelle varie distribuzioni di Linux)

Come installare R

1. Tramite il vostro browser visitate il sito
   http://www.r-project.org
   e cliccate sul link CRAN situato nella sezione Download
2. Scegliete un Mirror vicino a voi, ossia un qualsiasi sito italiano (consiglio: Padova)
3. Scegliete il vostro sistema operativo (Windows, Mac OS X, Linux)
   - **3.1** per Windos: cliccate su Base,
     cliccate su *R-2.11.1-win32.exe*,
     salvate il file eseguibile sul vostro computer
   - **3.2** per Mac OS X 10.5: cliccate su *R-2.11.1.pkg*
     per sistemi precedenti: clicca su *old*,
     cercare la versione di **R** compatibile
   - **3.3** per Linux ditribuzione Ubuntu é consigliabile usare il programma Synaptic Package Manager e inserire la voce 'R cran' nel motore di ricerca

Un aspetto importante da notare é che con la precedente procedura si é installato:

- il motore di calcolo
- il motore grafico del programma
- un insieme di routines che costituiscono il pacchetto base

Oltre a questo pacchetto base esistono altri pacchetti aggiuntivi. Questi pacchetti aggiuntivi sono stati 'scritti' dalla comunitá scientifica, ed in questo senso é importante avere una grande numero di users. Questo in altre parole significa che per la maggior parte dei modelli statistici utilizzabili esistono giá le specifiche routines da lanciare in **R**.

Istallazione dei pacchetti aggiuntivi

Per installare i pacchetti aggiuntivi si puó seguire la seguente procedura dal menu a tendina:

1. cliccate su Pacchetti
2. scegliete un Mirror
3. cliccate su Installa pacchetto...
4. fate il download del pacchetto scelto

Ora apriamo **R**!

Il messaggio iniziale della console di **R** si conclude con il simbolo $>$ che, in gergo, viene definito prompt dei comandi. Questo simbolo segnala all'utente che **R** é pronto ad eseguire il comando digitato dopo di esso. Un comando di **R** é sempre concluso dal tasto Invio [Return].
Iniziamo a giocare con la sintassi dei comandi di **R** attraverso semplici esempi.

Primi passi con R

```
> 2+2
[1] 4
> 2*3
[1] 6
> 3/5
[1] 0.6
> 3 ^ 3
[1] 27
> # questo é solo un commento
```

Si puo notare come il risultato del comando sia preceduto da $[1]$ che sta ad indicare che l'output é un vettore di lunghezza 1. La spiegazione di questa situazione é che **R** lavora per costruzione con i vettori, quindi interpreta un numero come un vettore di dimensione 1.
Il simbolo $\sharp$ serve per introdurre commenti nelle routines **R**.

Gli operatori di assegnazione

Possiamo effettuare assegnazioni in due modi:

- l'operatore $<-$ ovvero il tasto minore seguito dal tasto meno senza uno spazio tra i due simboli
- l'operatore $=$

$>$ a

Errore: oggetto "a" non trovato

$>$ b

Errore: oggetto b non trovato

$>$ a $<-$ 4

$>$ a

$[1]$ 4

$>$ b=5

$>$ b

$[1]$ 5

$>$ c=a+b

$[1]$ 9

Tipi e oggetti

In **R** un oggetto, definito attraverso una operazione di assegnazione, puó essere di tre tipi:

- un elemento numerico
- un elemento composto da una stringa di caratteri
- un elemento logico

Oggetti Numerici

Gli elementi numerici a loro volta si possono distinguere in interi, reali e complessi. Un facile esempio é dato da:

a=10

b=66.33

c=3+2.3i

Precisazione. L'insieme dei numeri rappresentabili é in realt á un sottoinsieme dei numeri naturali e reali. Un altra puntualizzazione é che i numeri reali sono approssimati attraverso un insieme finito di razionali. In **R**, comunque, non ci sono particolari problemi di approssimazione nelle applicazioni pratiche in quanto il piú piccolo numero rappresentabile é dell'ordine di 10^{-300}

Gli altri tipi

Per definire un oggetto come stringa di caratteri é necessario virgolettare la stringa, ad esempio:

d= ”Ciao”

e=”estate”

Il terzo tipo é formato dagli operatori logici TRUE e FALSE o, pi ú semplicemente, T e F. Questi operatori logici si utilizzano, ingenere, all’interno di funzioni come operatori di controllo.

f=TRUE

g=FALSE

h=F

i=T

Usiamo `mode(...)` per ottenere stampato sulla console il modo dell’oggetto

```
> mode(a)
[1] numeric
```

I Vettori

Fino ad ora attraverso l'operazione di assegnazione si sono definiti degli oggetti composti da un unico elemento. Per definire un oggetto composto da piú elementi é necessario utilizzare il comando di concatenamento `c(...)`.
In pratica per definire un vettore numerico bastera inserire come argomento di tale funzione i singoli elementi numerici separati da una virgola, mentre per definire un vettore di stringhe occorre ricordarsi, inoltre, di virgolettare gli elementi, come ad esempio...

```
>vet1=c(1,2,3)
>vet1
[1] 1 2 3
>vet2=c("a","b","c")
>vet2
[1] a b c
>vet3=c(a,b,c)
> vet3
[1] 10.00+0.0i 66.33+0.0i 3.00+2.3i
>vet4=c(f,g,h,i)
> vet4
[1] TRUE FALSE FALSE TRUE
> # i vettori possono anche essere vuoti (0 elementi)
> vet5=c()
> vet5
NULL
> # Attenzione vet1*vet1 fa il quadrato dei singoli elementi
> vet1*vet1
[1] 1 4 9
> # Comando per conoscere la lunghezza di un vettore
> length(vet1)
[1] 3
```

```
> # concatenare i vettori
> vet6=c(vet1,vet2)
> vet6
[1] "1" "2" "3" "a" "b" "c"
> # conversione vettori
>vet7=as.numeric(vet4)
>vet7
[1] 1 0 0 1
> # la conversione  avvenuta secondo la regole TRUE=1 e FAlSE=0
>vet8=as.character(vet1)
>vet8
[1] 1 2 3
```

Un po' di accortezza va posta sul fatto che i vettori devono essere composti da elementi omogenei, ovvero appartenenti tutti allo stesso tipo.

Talvolta é necessario modificare un elemento del vettore lasciando invariato tutto il resto. In **R** l'i-esimo elemento di un vettore generico a é dato da: a$[i]$. Quindi per modificare questo elemento é sufficiente effettuare una nuova operazione di assegnazione, come ad esempio:

$>$ a=c(1,2,3)

$>$ a

$[1]$ 1 2 3

$>$ a[2]=77

$>$ a

$[1]$ 1 77 3

Spesso in statistica é necessario dover definire particolari vettori composti o da sequenze numeriche o solamente da uno stesso elemento. Per evitare di imputare manualmente questi vettori si possono utilizzare le funzioni `rep(...)` e `seq(...)`.

Il comando `rep(x,n)` crea un vettore ripetendo un oggetto x per n volte, ad esempio

> b=c(1,2)

>d=rep(b,3)

> d

[1] 1 2 1 2 1 2

> e=rep(1,6)

> e

[1] 1 1 1 1 1 1

Il comando `seq(start, end, by=)` crea in un vettore una sequenza di numeri. I parametri start ed end sono rispettivamente il punto di inizio ed il punto di fine della sequenza, mentre by rappresenta il passo della sequenza. Giusto per dare un esempio del comando abbiamo che

```
> seq(1,15,3)
[1] 1 4 7 10 13
> 1:100
[1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
[16] 16 17 18 19 20 21 22 23 24 25 26 27
[31] 31 32 33 34 35 36 37 38 39 40 41 42
[46] 46 47 48 49 50 51 52 53 54 55 56 57
[61] 61 62 63 64 65 66 67 68 69 70 71 72
[76] 76 77 78 79 80 81 82 83 84 85 86 87
[91] 91 92 93 94 95 96 97 98 99 100
```

L'ultimo comando digitato rappresenta una scorciatoia nel caso in cui il passo della sequenza sia uguale ad uno.

Le Matrici

Fino ad ora abbiamo definito dei vettori mentre ora introdurremo le matrici. Il comando é `matrix(ogg, nrow = , ncol =)`, ogg é un generico oggetto mentre `nrow` e `ncol` sono i parametri che permettono di fissare il numero di righe ed il numero di colonne della matrice da creare. Proviamo a descrivero il comando matrix da qualche esempio.

```
> m1=matrix(ncol=2,nrow=2)
> m1
     [,1][,2]
[1,] NA NA
[2,] NA NA
>m2=matrix(0,ncol=2,nrow=2)
>m2
     [,1][,2]
[1,]  0   0
[2,]  0   0
>m3=matrix(1:9,ncol=3,nrow=3)
> m3
     [,1][,2][,3]
[1,]  1   4   7
[2,]  2   5   8
[3,]  3   6   9
m4=matrix(1:9,ncol=3,nrow=3,byrow=T)
>m4
     [,1][,2][,3]
[1,]  1   2   3
[2,]  4   5   6
[3,]  7   8   9
```

Come si vede dal primo esempio, non avendo specificato l'oggetto con cui "riempire" la matrice, gli elementi risultano missing ovvero `NA`.
Per accedere ad un elemento di una matrice é sufficiente specificare tra parentesi quadrate la riga e la colonna dellelemento separandole con una virgola. Se si vuole accedere alla i-esima riga di una matrice A si deve digitare A$[i,]$, analogamente per la j-esima colonna di A si deve digitare A$[,j]$.

```
> A=matrix(1:4,ncol=2)
>A
     [,1][,2]
[1,]    1    3
[2,]    2    4
> A[1,]
[1]  1  3
>A[1,2]
[1]  3
> A[,2]
[1]  3  4
```

Attenzione: potrebbe accadere una situazione del tipo:

$>$ A=matrix(1:5,ncol=2,nrow=2)

```
Warning message:
la lunghezza [5] dei dati non é un sottomultiplo o un multiplo
del numero di righe [2] in matrix
```

$>$ A

```
     [,1][,2]
[1,]  1   3
[2,]  2   4
```

Abbiamo fatto un errore (mancata coerenza nelle dimensioni dell'oggetto rispetto a quelle della matrice) nella definizione della matrice ma questa é stata comunque creata.

Talvolta é necessario fare alcune operazioni con le matrici come:

- conoscere le dimensioni
- fusioni di due matrici
- trasposte
- inverse
- prodotti righe per colonne
- ...

Per le dimensioni si possono usare i seguenti comandi

$>$ dim(A)

[1] 2 2

$>$ nrow(A)

[1] 2

$>$ ncol(A)

[1] 2

Per le fusioni di matrici vi sono i comandi `rbind(...)` e `cbind(...)`.
Il primo unisce due matrici o vettori (attenzioni alle dimensioni che potete controllare con il comando `dim(...)`) per le righe, in pratica mette una matrice sopra l'altra.
Il secondo comando opera la fusione, invece, affiancando orizzontalmente le matrici.
Due esempi chiarificheranno il tutto.

```
> a=matrix(1:6,ncol=2)
> a
     [,1][,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> b=7:9
> b
[1]  7  8  9
> prova=cbind(a,b)
> prova
[1,] 1 4 7
[2,] 2 5 8
[3,] 3 6 9
> c=c(1,2)
> prova2=rbind(a,c)
> prova2
[1,] 1 4
[2,] 2 5
[3,] 3 6
[4,] 1 2
```

Passiamo ora alle altre operazioni con le matrici.

- `t(...)` che implementa la trasposta di una matrice
- `solve(...)` codifica l'inversa
- $\%*\%$ é l'operatore che produce il prodotto righe per colonne. Un consiglio utile é quello di effettuare il prodotto righe per colonne solamente tra matrici, cio é definire anche i vettori con il comando matrix per evitare ambiguitá tra le dimensioni.

data.frame

Utili per analisi statistiche poiché possono contenere elementi di natura diversa.

> nome=c(”Alessandra”,”Luca”,”Anna”,”Paolo”)

> eta=c(25,40,5,2)

> sesso=c(”F”,”M”,”F”,”M”)

> vaccino=c(T,T,F,F)

i vettori precedenti possono essere messi tutti in un’unica struttura

> dati=data.frame(nome,eta,sesso)

> dim(dati)

> dati=cbind(dati,vaccino) aggiungo colonna

> nome=c(”Sara”,”Maria”)

> eta=c(26,30)

> sesso=c(”F”,”F”)

> vaccino=c(T,T)

> dati2=data.frame(nome,eta,sesso,vaccino)

> dati=rbind(dati,dati2)

> dati

> dim(dati)

ordinamento delle osservazioni

$>$ sort()

$>$ order() ♯ mi indica la posizione

per riorganizzare data.frame secondo chiavi di ordinamento

$>$ dati[order(dati$sesso, dati$eta),]

Caricamento dati

Per leggere i file esterni esistonocomandi diversi a seconda del tipo di files

- file Excel: si puó utilizzare il comando `read.xls('nomefile.txt')`; questo comando potrebbe non funzionare poiché necessita di applicazioni perl. Si consiglia di salvare il file in formato .csv e quindi leggere il file con il comando `read.csv('nomefile.csv')`
- file formato ASCHII: posso usare uno dei seguenti comandi:
  `read.table()` legge un file memorizzato su disco, inserendo i dati direttamente in un data frame
  `read.csv()`;
  `read.fwf()`;
  `scan()` legge un file di input e memorizza i dati in un vettore o una lista

read.table

Come si usa ”read.table”:
`read.table('nomefile.formato', header = TRUE/FALSE, sep = '' '', dec = ''.'', skip=1)`
Una serie di comandi separati da virgole:

- si inserisce il nome del file tra virgolette
- `header` indica se sul data set é presente il nome delle variabili (TRUE se si, FALSE se no)
- `sep` indica come i diversi dati sono separati tra loro (es. nt se c'é uno spazio)
- `dec` indica come sono definiti gli eventuali numeri decimali
- `skip` indica il numero di righe da non importare iniziando dalle prime

facciamo un sempio

> dati=read.table('cars.txt' , header = TRUE, sep =" ")
♯ possiamo anche usare
>dati=read.csv('cars.csv', header = TRUE, sep =" ")
>scan('cars.csv',skip=1,sep= " ") ♯ devo saltare il nome delle variabili

Statistiche

Cominciamo a fare alcune semplici statistiche sui dati.

> summary(dati)
♯ piccolo sommario statistico, fornisce i quartili, la media, e il range.

> range(dati$speed)

> med1=median(dati$speed)

> m1 = mean(dati$speed)

> var1= var(dati$speed)

♯ la dipendenza tra due variabili puó essere verificata osservendo la covarianza
> c = cov(dati$speed,dati$dist) ♯ person di default

♯ per verificare se la dipendeza riscontrata é lineare calcoliamo la correlazione
> cr = cor(dati$speed,dati$dist)

> table(dati$speed)
♯ con questo comando otttengo la distribuzione di frequenza assoluta

♯ il comando `length` conta il numero di ossevazioni, quindi

> table(dati$speed)/length(dati$speed)

♯ rapporta le frequenze assolute alla numerositá fornendo le frequenze relative!

Il Boxplot

Il comando boxplot é molto semplice

> boxplot(dati$speed)

> boxplot(dati$speed, border="red", col="light grey", horizontal=TRUE)

Come si legge il Boxplot?

1. La linea centrale evidenzia la mediana
2. Sopra e sotto la mediana ci sono il primo e il terzo quartile
3. la larghezza della scatola é detta intervallo o scarto interquartile SIQ = Q3 - Q1. Nel SIQ si posizionano almeno il 50% dei valori
4. Le linee al termine dei baffi sono poste a Q1 - 1.5SIQ (a sinistra) e Q3 + 1.5SIQ (a destra)
5. Le osservazione esterne alle linee che terminano il baffo sono indicate da pallini (potenziaili outliers)
6. Se non ci valori fuori dai baffi, il baffo termina nel valore intero precedente (successivo) alla osservazione piú piccola (piú grande), come nel nostro caso (baffo a sinistra)

Il box plot evidenzia come l'istogramma l'asimmetria della distribuzione.

Salvare i grafici

Quando si elabora un grafico, questo appare in una nuova finestra. Cliccando in alto a sinistra su File appare Salva con nome, puntandoci sopra il mouse appaiono tanti modi diversi di salvare il grafico: postscript, pdf, bmp, jpeg... Attenzione! Se facciamo un altro grafico si sovrapporrá al primo: se vogliamo tenerceli é importante salvarli subito. Per inserire il grafico in un documento, ad esempio Word, basta cliccare sul grafico col tasto destro e poi sinistro su copia come bitmap oppure copia come metafile e poi incolla sul documento aperto.

```
a=c(0.1, 0.3, 0.25, 0.15, 0.2)
pie(a)
b=c(0.2, 0.2, 0.2, 0.3, 0.1)
pie(b)
```

Istogramma

Quando la variabile é (virtualmente) continua o discreta ma con un elevato numero di modalitá (le misurazioni sono sempre discrete!), tabelle o grafici che riportino tutti i valori distinti e le relative densitá possono essere poco sintetiche o informative Negli istogrammi

- l'asse orizzontale é un segmento dellasse reale suddiviso in intervalli (classi)
- l'area dei rettangoli (barre) rappresenta la frequenza relativa
- l'area totale dell'istogramma é uguale 1

Il comando `hist()`

- suddivide la distribuzione in classi, determinandone numero ed estremi
- calcola le frequenze dei casi che ricadono in ciascuna classe
- procede alla generazione del grafico corrispondente

Alcuni esempi

$>$ hist(dati$speed)

$>$ hist(dati$speed, prob=TRUE, border="grey", col="light grey", xlab=" ", ylab=" ", main="speed")

$>$ lines(density(dati$speed), col="royalblue", lwd=2) ♯ approssimazione

Help

Cominciamo dai comandi di help in linea. Con questo strano nome vengono definiti i comandi di aiuto che possono lanciati essere dal prompt. In particolare in **R** sono a disposizione dellutente tre diversi comandi:

- help(...)
- help.search(...)
- help.start()

Con il primo comando **R** apre una nuova finestra in cui spiega la sintassi del comando, descrive gli argomenti e fornisce degli esempi su come utilizzare tale comando, ad esempio provate a digitare `help(quit)` o, in maniera piú rapida, `?quit` e studiate la finestra che esce in pop-up.
Questo comando di help viene utilizzato quando l'utente conosce il nome di un comando che intende utilizzare.

Quando non si conosce il nome di un comando da utilizzare, si puó effettuare una ricerca per parole chiavi attraverso l'uso del `help.search(''...'')`.
Questo comando apre una finestra (eventualmente vuota) in cui sono presenti tutti i riferimenti dove le vostra stringa di parole chiavi é presente.

Supponiamo, ad esempio, che non si conosca il comando del coseno, allora, il primo passo é quello di digitare `help.search(cos)`.

```
Help files with alias or concept or title matching cos using
regular expression matching:
cosh(base) Hyperbolic Functions
Trig(base) Trigonometric Functions
Airline(Ecdat) Cost for U.S. Airlines
Electricity(Ecdat) Cost Function for Electricity Producers, 1970
...
```

Nella finestra che si apre al secondo punto della lista compare un comando Trig (ATTENZIONE: **R** é case sensitive ovvero distingue fra maiuscole e minuscole) che si trova nel pacchetto base che contiene le funzioni trigonometriche.
A questo punto il passo successivo della ricerca é digitare `help(Trig)` scoprendo che il coseno di x é semplicemente `cos(x)`.

Il piú flessibile ed intuivo ma anche il piú dispendioso in termini di tempo é il comando `help.start()`.
Digitando questo comando si apre una finestra di help interattiva tramite il vostro browser di default (IE, Firefox, Opera, ...). Cliccando sui vari link potete sfogliare il manuale che accompagna **R**. Questo manuale si trova sull hard disk del vostro pc (C:/Programmi/R/R-2.10.1/doc/manual/), quindi é consultabile anche quando non siete on-line.

Tutti e tre i comandi sono utilizzabili anche tramite i menu a tendina di **R** sotto la voce Aiuto. Lultimo comando di help richiama, in realtá, l'accesso elettronico ai manuali che si dispongono nelle sottocartelle della cartella dove **R** é stato installato.

Documentazione On-Line

Maggiori riferimenti bibliografici e link a documentazione di varia natura possono essere trovati sul sito ufficiale, all'indirizzo http://www.r-project.org/other-docs.html.

Salvataggio e Memoria

Un aspetto importante da conoscere é come uscire da **R** salvando il proprio lavoro. Innanzitutto il comando per uscire da una sessione é `quit()` o, piú semplicemente, `q()`.
Una volta mandato in esecuzione questo comando, il sistema apre in pop-up una finestra che chiede il salvataggio dell'area di lavoro

- Rispondendo no si ottiene la chiusura del programma senza alcun salvataggio
- Rispondendo si, invece, il sistema salva tutti gli oggetti ed i comandi presenti in memoria (workspace) e termina la propria sessione. Se si riapre **R** dopo un salvataggio, il sistema ricarica automaticamente in memoria l'ultimo workspace.

Salvataggio e Memoria - 2

Il salvataggio dellarea di lavoro consiste nella creazione di due files nella directory di **R**: `.Rhistory` e `.Rdata`.
Il primo file é un file di tipo ASCII che puó essere aperto tramite un qualunque editor di testo (es. Blocco Note) e contiene tutti i comandi eseguiti durante la sessione salvata.
Il secondo é, invece, un file di tipo binario (codificato in linguaggio macchina e, quindi, dal contenuto illeggibile).

Salvataggio e Memoria - 3

Un comando che ci permette di visionare l'insieme di oggetti contenuti nella memoria del programma é dato dal comando `ls()`. Tale comando restituisce in uscita un vettore composto dai nome di tutti gli oggetti in memoria. Se non si é ancora definito alcun oggetto restituisce l'espressione `character(0)`, ovvero un vettore composto da 0 elementi.
Se si desidera rimuovere un oggetto dalla memoria, ad esempio perché ne occupa troppa, si puó utilizzare il comando `rm(nomeoggetto)`. Se si desidera rimuovere piú di un oggetto allora bisogna inserire come argomento della funzione una sequenza di nomi separati da una virgola oppure un vettore di tipo character. Per rimuovere tutti gli oggetti dal dataset é necessario eseguire una variante del precedente comando data da: `rm(list=ls(all=TRUE))`.

Salvataggio e Memoria - 4

E' utile scegliere la directory di lavoro (working directory o WD) in cui si andrá a salvare il workspace ed, eventualmente modificarla.
I comandi che assolvono a queste necessitá sono: `getwd()` e `setwd(...)`.
Il primo fornisce in output il percorso (path) dellattuale directory di lavoro.
Il secondo comando permette, invece, di cambiare WD specificando all'interno delle parentesi il path della directory scelta.