STATISTICA 1, metodi matematici e statistici

Introduzione al linguaggio R

Esercitazione1: 25-02-2005

Luca Monno

Università degli studi di Pavia

luca.monno@unipv.it
http://www.lucamonno.it

Materiale di riferimento

An Introduction to R : `www.r-project.org`

S.M. Iacus, G. Masarotto. (2002) Laboratorio di Statistica con R . MacGrow Hill. Milano

Materiale on line per R (Suggerimenti)

An Introduction to R : `www.r-project.org` Manuals

Verosimiglianza ed R : `www.stat.unipd.it/sartori` Appunti per i dottorandi

Laboratori di statistica matematica a Roma 3
`www.mat.uniroma3.it/didatticacds/corsi/didattica_interattiva/aa_01_02/st1/st1.html`

Practical regression and Anova with R `www.r-project.org` Contributed

Per iniziare

Prima di iniziare create una directory con su D

Per entrare in R dal desktop andate su Start/Programmi/R e cliccate su l'icona con la R

Sullo schermo apparirà una finestra bianca riconoscibile per la presenza del prompt da cui si potranno scrivere i comandi.

Programma della prima esercitazione

Che cosa è R e come utilizzarlo

Operazioni algebriche, vettori e matrici

Gli oggetti di R

Funzioni, test e cicli.

Che cosa è R e come utilizzarlo

Cosa è R

R is a language and environment for statistical computing and graphics... ovvero

R è un ambiente integrato che permette di elaborare dati, eseguire calcoli ed effettuare rappresentazioni grafiche

- Si compone di una serie di strumenti per l'analisi statistica dei dati;
- si tratta di un linguaggio pensato per descrivere modelli statistici anche estremamente complessi;
- permette la rappresentazione grafica di dati;
- è un linguaggio *objectd-oriented* che può essere facilmente esteso dall'utente finale;
- è gratuito e *Open source*, ovvero ciascuno può avere accesso al codice interno di R ;

Come interagire con R

Si interagisce con R attraverso la R -console digitando dopo il prompt “>” i comandi che vogliamo eseguire. Ad esempio digitando

```
>help(plot)
```

otterremo l’aiuto in linea per il comando `plot`, mentre con il comando

```
>demo(graphics)
```

si ottiene una dimostrazione delle potenzialità grafiche di R .
Per uscire da R basta digitare

```
>q()
```

... e rispondere alla seguente domanda:

```
Save workspace image?  [y/n/c]:
```

Spazio di lavoro (workspace) e dintorni

Uscendo da R possiamo quindi salvare un immagine del **workspace**.
Nel **workspace** abbiamo l'insieme degli oggetti che vengono creati durante le varie sessioni di lavoro.
Salvando il workspace, alla successiva utilizzazione di R , questo verrà caricato automaticamente insieme al file `.Rhistory` ovvero la sequenza di comandi (corretti e sbagliati) che abbiamo eseguito durante la sessione di lavoro.

Operazioni algebriche, vettori e matrici

R lavora in generale con dati strutturati e le strutture più semplici sono scalari e vettori. Ad esempio

```
> x <- 4
```

assegna ad x il valore 4. Se invece vogliamo creare un vettore (inteso come una stringa di valori consecutivi) y il comando è

```
> y <- c(4, 2, 7, 2)
```

Il comando `c` serve per concatenare gli elementi che vengono forniti come argomento: osserviamo per esempio che

```
> c(x, y)
```

```
[1] 4 4 2 7 2
```

R esegue operazioni sia vettoriali che scalari. Se diamo il comando

```
> x * y
```

```
[1] 16  8 28  8
```

otteniamo prodotto di uno scalare per un vettore. Se invece scriviamo

```
> y * y
```

```
[1] 16  4 49  4
```

otteniamo il vettore composto dagli elementi 16,4,49,4, cioè il prodotto termine a termine degli elementi di `y` con se stessi.

In generale applicando gli operatori `+ - * / ^` a due vettori numerici otteniamo un vettore contenente il risultato dell'operazione fatta elemento per elemento. Gli elementi del vettore più corto vengono reiterati quando necessario con un warning se la lunghezza del vettore più lungo non è multipla di quella del vettore più corto.

Il prodotto righe per colonne (utilizzato nell' algebra lineare) si ottiene invece atttaverso l'operatore %*%. In questo caso R interpreta i vettori come vettori colonna,

```
> t(y) %*% y
```

produce quindi il prodotto $y^t \times y$. (Il comando `t` opera la trasposzione del vettore)

Per creare delle matrici si utilizza il comando `matrix`. Ad esempio

```
> a <- c(1, 2, 3, 4, 5, 6)
> A <- matrix(a, nrow = 3, ncol = 2)
```

crea una matrice di dimensione 2×3 utilizzando gli elementi del vettore `a` e riempiendo sequenzialmente prima la prima colonna e poi la seconda. Se invece vogliamo riempire la matrice per righe basta passare l'opzione `byrow=T`, ovvero

```
> B <- matrix(a, nrow = 3, ncol = 2, byrow = T)
```

E’ possibile eseguire un operazione termine a termine anche tra matrici, quando queste hanno la stessa dimensione. Ponendo ad esempio

```
> x <- matrix(c(1, 2, 3, 4), 1, 4)
> y <- matrix(c(1, 2, 3, 4), 4, 1)
```

il prodotto

```
> x * t(y)

     [,1] [,2] [,3] [,4]
[1,]    1    4    9   16
```

produce effettivamente il prodotto termine a termine di `x` per `y`. Mentre

```
> x * y
```

produce un messaggio di errore.

Generazione di successioni regolari

Per generare delle sequenze regolari possiamo utilizzare l'operatore `:` oppure il comando `seq`

```
> s1 <- 1:30
> s2 <- seq(0, 1, by = 0.1)
> s3 <- seq(0, 1, length = 10)
```

Nel comando `seq` i primi due argomenti specificano il primo e l'ultimo valore della successione, con `by` specifichiamo il passo mentre con `length` la lunghezza della successione.

Un comando simile è `rep` che serve a replicare lo stesso oggetto più volte, ad esempio

```
> s4 <- rep(s2, times = 5)
```

produce un vettore con 5 volte il vettore `s2`.

Accedere agli elementi di vettori e matrici

Possiamo individuare un generico elemento (diciamo il terzo) di un vettore x

```
> x <- c(2, 4, 1, 2, 4)
```

con il comando

```
> x[3]
```

```
[1] 1
```

e una colonna (diciamo la seconda) di una matrice B con

```
> B[2, ]
```

```
[1] 3 4
```

I sottoinsiemi degli elementi di un vettore possono essere selezionati appendendo al nome del vettore un vettore di indici tra parentesi quadre. Possibili vettori di indici sono quelli composti da interi positivi

```
> x[1:4]

[1] 2 4 1 2

> x[c(2, 3)]

[1] 4 1
```

Esempi di vettori di indici sono anche i vettori logici. Ponendo

```
> g <- x < 3
```

abbiamo che g è un vettore che assume valore logico `TRUE` in corrispondenza degli elementi di `x` minore di 3 e `FALSE` negli altri. Allora con

```
> x[g]
```

```
[1] 2 1 2
```

otteniamo il sottovettore di `x` composto solo dagli elementi minori di 3. Vettori di indici sono anche vettori con interi negativi

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

In questo caso il risultato sarà un vettore senza gli elementi corrispondenti agli indici specificati.

Gli oggetti di R

Il workspace

Tutti gli oggetti che abbiamo creato possono essere visualizzati digitando

```
> ls()
```

Possiamo salvare il contenuto attuale del workspace con il comando

```
> save.image()
```

Il nome del file è per convenzione `.RData` e viene memorizzato nella directory corrente di lavoro a cui possiamo risalire attraverso

```
> getwd()
```

e che possiamo cambiare attraverso

```
> setwd("D://Rsti")
```

Per salvare solo alcuni oggetti del workspace, ad esempio `x` e `B` possiamo utilizzare il comando `save` specificando la lista degli oggetti da salvare

Ogni cosa in R è un oggetto. Possibili tipi di oggetti sono: `character`, `numeric`, `integer`, `logical`, `complex`

Tutti questi oggetti possono essere scalari o matrici a due o più vie. Possiamo avere informazioni su che tipo è un determinato oggetto attravero il comando `mode` che ci dice di che tipo sono le parti da cui è costituito un oggetto.

```
> mode(B)
> mode(c("master"))
> C <- rbind(B, c(1, "master"))
> mode(C)
> C
```

Quando un vettore od una matrice ha un elemento di tipo `character` tutte le loro parti vengono considerate `character`

Esiste poi il concetto di lista che può essere visto come un contenitore di oggetti dei più disparati.

```
> L <- list(L1 = C, L2 = c(1, 2))
> mode(L)
```

Per estrarre gli elementi da una lista possiamo richiamre il nome della lista seguito da $ più il nome dell'elemento, oppure specificando l'indice dell'elemento tra due parentesi quadre

```
> L$L1
> L[[1]]
```

Informazioni sul tipo degli elementi della lista possono essere ottenuti con il comando `str`

```
> str(L)
```

Altri tipi di oggetti

Oltre alle liste esistono altre classi di oggetti con una struttura diversa da quella dei vettori e delle matrici

- array
- fattori
- data frames
- funzioni

Funzioni, test e cicli

Funzioni

In R è possibile creare della funzioni personalizzate. La sintassi generale è

`nomefunzione<-function(`$\texttt{arg}_1$`,`$\texttt{arg}_2$`,...)  espressione`

Ad esempio

```
> somma <- function(a, b) a + b
```

Il nome della funzione è `somma` ed è una funzione di due argomenti. L'espressione da valutare è la somma dei due oggetti. Tali oggetti devono appartenere ad una classe all'interno della quale l'operazione di somma è definita.

```
> somma(2, 3)
> somma("roma", "lazio")
```

i test

R permette di utilizzare due strumenti fondamentali per un linguaggio di programmazione: i cicli e i test (logici). La costruzione di un test viene fatta nel modo seguente

```
>if (condizione) conseguenza1 else conseguenza2
```

Vediamone un esempio all'interno di una funzione

```
> rulette <- function(numero, euro) {
+     r <- sample(x = seq(0, 36), size = 1)
+     if (r == numero)
+         vincita <- 36 * euro
+     else vincita <- 0
+     return(vincita)
+ }
```

Gli operatori logici sono ==, != (disuguaglianza), <, >, <=, >=, & (intersezione), | (unione).

I cicli

Eseguire un ciclo in R è molto semplice.

```
> for (i in 1:10) print(rulette(10, i))
```

Bisogna notare che i cicli in R sono molto dispendiosi in termini di tempo macchina e il più delle volte possono essere evitati. Consideriamo infatti il seguente esempio

```
> a <- 0
> for (i in 1:10000) a <- a + i
> a

[1] 50005000

> sum(1:10000)

[1] 50005000
```