Materiale didattico per i laboratori di Modelli Statistici I

a cura di:
A. Brazzale, M. Chiogna, C. Gaetan, N. Sartori e L. Ventura

Anno Accademico 2002-2003

Indice

Capitolo 0

Richiami di R

0.1 Iniziare e chiudere una sessione di R

Per iniziare una sessione R fare un doppio click di mouse sulla icona di R.
Per uscire da R, usa `q()`. Per salvare i dati rispondere "Si", altrimenti rispondere "No".
Per controllare cosa c'e' disponibile nella directory dei dati:

```
> ls()
character(0)
```

Per eliminare un oggetto, usa `rm()`.

```
> rm(thing)
> thing
Error: Object "thing" not found
```

Se si vogliono eliminare più oggetti, bisogna elencarli separati da virgole.

```
> rm(thing1,thing2)
```

Quando si inizia una nuova sessione di lavoro, è opportuno rimuovere tutti i vecchi oggetti che non servono. Un comando utile è:

```
> rm(list=ls())      oppure     rm(list=objects())
```

0.2 Semplice aritmetica

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 ** o ^ per calcolare un elevamento a potenza.
[1] 108
```

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

```
> sqrt(2)
[1] 1.414214

> sin(3.14159)          sin(Pi greco) e' zero
[1] 2.65359e-06         e questo e' vicino...
```

Fornisce anche il valore di π

```
> sin(pi)
[1] 1.224606e-16    ancora piu' vicino a  zero...
```

Ecco una breve lista

```
  Nome                  Operazione

  sqrt                                  radice quadrata
  abs                                   valore assoluto
  sin   cos   tan               funzioni trigonometriche
  asin  acos  atan      funzioni trigonometriche inverse
  exp   log             exponenziale e logaritmo naturale
```

Le funzioni possono essere annidate:

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

0.3 Assegnazioni di valori

Si può salvare un valore assegnandolo ad un oggetto mediante il simbolo `<-` , oppure il simbolo `_` , o anche il simbolo `=`

```
> x <- sqrt(2)      salva in x la radice quadrata di 2
> x
[1] 1.414214
> x**3
[1] 2.828427
```

Valori logici
R permette di gestire operazioni e variabili logiche:

```
> x <- 10          fissa x uguale a 10
> x > 10           x e' piu' grande di 10?
[1] FALSE
> x <= 10
[1] TRUE
> tf <- x > 10
> tf
[1] FALSE
```

0.4 Gestione di vettori

0.4.1 Creazione di vettori

Per creare un vettore, si usa la funzione `c()`:

```
> x <- c(2,3,5,7,11)
> x
[1]  2  3  5  7 11
```

Se si hanno tanti dati da scrivere, puo' essere più conveniente usare `scan()`:

```
> x <- scan()
1: 1
2: 6
3: 3
4: 4
5:
> x
[1] 1 6 3 4
>
```

```
> x <- scan()
1: 23 34 32
4: 33 88 44
7:
```

Esercizio: `scan()` può anche servire per leggere un vettore da un file. Con un editor, prova a creare il file data1.dat contenente i seguenti dati:

```
243 251 275 291 347 354 380 392
206 210 226 249 255 273 289 295 309
241 258 270 293
```

Puoi leggere il vettore con il comando:

```
> redcell <- scan("data1.dat")
```

Successioni
Si può usare la notazione a:b per creare vettori che sono sequenze di numeri:

```
> xx <- 1:10
> xx
 [1]  1  2  3  4  5  6  7  8  9 10

> xx <- 100:1
> xx
  [1] 100  99  98  97  96  95  94  93  92  91  90  89  88  87  86  85  84 83
 [19]  82  81  80  79  78  77  76  75  74  73  72  71  70  69  68  67  66 65
 [37]  64  63  62  61  60  59  58  57  56  55  54  53  52  51  50  49  48 47
 [55]  46  45  44  43  42  41  40  39  38  37  36  35  34  33  32  31  30 29
 [73]  28  27  26  25  24  23  22  21  20  19  18  17  16  15  14  13  12 11
 [91]  10   9   8   7   6   5   4   3   2   1
```

La stessa operazione può essere fatta con:

```
> xx<-seq(from=100, to=1)
> xx
```

Possono anche essere creati dei vettori che contengono elementi ripetuti

```
> rep(2,times=3)
[1] 2 2 2
> rep(2,3)
[1] 2 2 2
> a_c(rep(2,3),4,5,rep(1,5))
> a
 [1] 2 2 2 4 5 1 1 1 1 1
```

Ai vettori puoò essere applicata la stessa aritmetica di base che è stata applicata ai valori scalari:

```
> x_1:10
> x*2
 [1] 2 4 6 8 10 12 14 16 18 20
> x * x
 [1] 1 4 9 16 25 36 49 64 81 100
```

Possono essere eseguite operazioni logiche anche sui vettori

```
> x > 5
 [1] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
```

0.4.2 Estrazione degli elementi da un vettore

Gli elementi di un vettore possono essere estratti usando le parentesi quadre []:

```
> xx[7]
[1] 94
```

Si possono estrarre anche sottoinsiemi di elementi:

```
> xx[c(2,3,5,7,11)]
[1] 99 98 96 94 90
> xx[85:91]
[1] 16 15 14 13 12 11 10
> xx[91:85]
[1] 10 11 12 13 14 15 16
> xx[c(1:5,8:10)]
[1] 100  99  98  97  96  93  92  91
> xx[c(1,1,1,1,2,2,2,2)]
[1] 100 100 100 100  99  99  99  99
```

Ovviamente, sottoinsiemi di elementi possono essere salvati in nuovi vettori:

```
> yy <- xx[c(1,2,4,8,16,32,64)]
> yy
[1] 100  99  97  93  85  69  37
```

Se le parentesi quadre racchiudono un numero negativo, l'elemento corrispondente viene omesso dal vettore risultante:

```
> x <- c(1,2,4,8,16,32)
> x
[1]  1  2  4  8 16 32
> x[-4]
[1]  1  2  4 16 32
```

Alcune funzioni utili per la manipolazione di vettori

```
> x <- 3:26
> length(x)
[1] 24                  ...il numero di elementi
> max(x)
[1] 26                  ...il massimo
> min(x)
[1] 3                   ...il minimo
> sum(x)
[1] 348                 ...la somma dei valori in x
> prod(x)
[1] 2.016457e+26   ...il prodotto dei valori in x
> mean(x)
```

```
[1] 14.5                ...la media aritmentica : sum(x)/length(x)
> var(x)
[1] 50                     ...la varianza corretta
> range(x)
[1]  3 26               ...il campo di variabilita'
```

0.5 Matrici

R consente anche di usare le matrici:

```
> x <- matrix(c(2,3,5,7,11,13),ncol=2)
> x
     [,1] [,2]
[1,]    2    7
[2,]    3   11
[3,]    5   13
```

NB: Bisogna specificare nrow o ncol per comunicare a R la dimensione della matrice. Se gli elementi di una matrice sono contenuti in un file, possiamo usare ancora `scan()`

```
1,24,32,36,33
2,16,44,34,33
3,20,31,43,32
4,23,35,37,35
5,27,40,40,31
6,19,43,32,37
```

Se questi elemeni sono contenuti nel file matdata, li possiamo mettere in una matrice 6X5 con il comando:

```
> x2 <- scan('matdata',sep=',')
> mx <- matrix(x2,ncol=5, byrow=T)
> mx
     [,1] [,2] [,3] [,4] [,5]
[1,]    1   24   32   36   33
[2,]    2   16   44   34   33
[3,]    3   20   31   43   32
[4,]    4   23   35   37   35
[5,]    5   27   40   40   31
[6,]    6   19   43   32   37
```

Per estrarre da una matrice un elemento, bisogna specificarne le due coordinate:

```
> x[2,1]
[1] 3
> x[2,2]
[1] 11
```

Se non si mette una delle coordinate, si ottiene una intera riga/colonna:

```
> x[,1]
[1] 2 3 5
> x[3,]
[1]  5 13
```

Possono essere estratti sottoinsiemi di righe e/o colonne:

```
> x <- matrix(1:16,ncol=4)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
> x[c(1,4),c(3,4)]          Riga 1 e 4,
     [,1] [,2]              Col  3 e 4
[1,]    9   13
[2,]   12   16
```

La funzione dim indica la dimensione (numero di righe e numero di colonne) della matrice

```
> dim(mx)
[1] 6 5
```

0.6 Data-frames

Un data frame è un oggetto simile ad una matrice, ma usato per rappresentare una matrice di dati. Ogni riga rappresenta una 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. Se il file `Cherry.dat` è così costituito:

```
8.3     70      10.3
8.6     65      10.3
8.8     63      10.2
10.5    72      16.4
10.7    81      18.8
10.8    83      19.7
...
...
```

possiamo acquisirlo con il comando:

```
> Ciliegi <- read.table("I:/modelli/Cherry.dat")
> Ciliegi                        (nota che Ciliegi e' diverso da ciliegi)
```

Il data frame è anche una matrice

```
> dim(Ciliegi)
[1] 31  3           (31 osservazioni e 3 variabili)
```

Se non specificati, i nomi delle tre variabili sono V1 V2 e V3:

```
> names(Ciliegi)
[1] "V1" "V2" "V3"
```

Si possono cambiare le etichette con il comando:

```
> names(Ciliegi) <- c('diametro','altezza','volume')
```

Alternativamente, si potevano assegnare questi nomi direttamente in fase di lettura da file:

```
> Ciliegi<-read.table("I:/modelli/Cherry.dat",
+ col.names=c("diametro","altezza","volume"))
```

Essendo il data frame una matrice, possiamo considerare, ad esempio, la terza variabile con:

```
Ciliegi[,3]
 [1] 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 24.2 21.0 21.4 21.3 19.1
[16] 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0
[31] 77.0
```

Tuttavia, la struttura di data frame permette un metodo migliore per indicare le variabili:

```
> Ciliegi$volume
 [1] 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 24.2 21.0 21.4 21.3 19.1
[16] 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0
[31] 77.0
```

Utilizziamo il comando `attach()` per comunicare ad R che le operazioni che faremo si riferiscono al dataframe `Ciliegi`:

```
> attach(Ciliegi)
> volume
 [1] 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 24.2 21.0 21.4 21.3 19.1
[16] 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0
[31] 77.0
```

Per avere delle statistiche di base sulle variabili contenute in `Ciliegi` possiamo usare la funzione `summary()`:

```
> summary(Ciliegi)
    diametro        altezza       volume
 Min.   : 8.30   Min.   :63   Min.   :10.20
 1st Qu.:11.05   1st Qu.:72   1st Qu.:19.40
 Median :12.90   Median :76   Median :24.20
 Mean   :13.25   Mean   :76   Mean   :30.17
 3rd Qu.:15.25   3rd Qu.:80   3rd Qu.:37.30
 Max.   :20.60   Max.   :87   Max.   :77.00
```

Possiamo anche rappresentare graficiamente la distribuzione di una variabile ad esempio `diametro`, mediante un istogramma

```
hist(diametro)
```

oppure un diagramma a scatola (*box–plot*)

```
boxplot(diametro)
```

Per estrarre elementi da un data frame valgono le stesse regole valide per le matrici.

```
> altezza
 [1] 70 65 63 72 81 83 66 75 80 75 79 76 76 69 75 74 85 86 71 64 78 80 74 72 77
[26] 81 82 80 80 80 87

> Ciliegi[altezza > 80,]                    estrae un dataframe...
   diametro altezza volume                 ...di alberi che..
5      10.7      81   18.8                 ...  sono alti piu' di
6      10.8      83   19.7                 ... 70 piedi
17     12.9      85   33.8
18     13.3      86   27.4
26     17.3      81   55.4
27     17.5      82   55.7
31     20.6      87   77.0
```

Capitolo 1

Modello lineare semplice

1.1 Analisi dei dati CHERRY.DAT

L'insieme di dati `Ciliegi` occorre acquisirlo da file:

```
> Ciliegi <- read.table("I:/modelli/Cherry.dat",
+ col.names=c('diametro','altezza','volume'))
```

Possiamo comunicare a R che le operazioni che faremo d'ora in avanti si riferiscono al data frame `Ciliegi`:

```
> attach(Ciliegi)
```

I dati contengono, per 31 alberi di ciliego abbattuti, la misura del volume di legno ricavato dall'albero (`volume`), il diametro del tronco misurato a circa un metro dal suolo (`diametro`) e l'altezza dell'albero (`altezza`). Vogliamo indagare la relazione tra il volume di legno e il diametro. Possiamo fare un grafico di `diametro` e `volume` con il comando:

```
> plot(diametro,volume)
```

Il numero di osservazioni è:

```
> n<-dim(Ciliegi)[1]
```

Calcoliamo le stime dei minimi quadrati della regressione `volume`$= \alpha + \beta *$ `diametro`:

```
> beta<-(sum(diametro*volume)/n-mean(volume)*mean(diametro))/
+ (mean(diametro^2)-mean(diametro)^2)
> beta
[1] 5.065856
```

Questo è equivalente a

```
>beta<-cov(diametro,volume)/var(diametro)
>beta
[1] 5.065856
```

La stima di α è quindi pari a

```
>alpha<-mean(volume)-beta*mean(diametro)
>alpha
[1] -36.94346
```

Aggiungiamo la retta stimata nel grafico, con il comando

```
> abline(alpha,beta,lty="dashed")
```

Il comando `abline(a,b)` traccia una retta nel grafico corrente con intercetta `a` e coefficiente angolare `b`. L'opzione `lty` è un'opzione grafica generale (valida ad esempio anche per il comando `plot`) e definisce il tipo di linea. Assume valori: "blank", "solid" (default), "dashed", "dotted", "dotdash", "longdash" o "twodash", oppure rispettivamente i numeri da 0 a 7. L'opzione "blank" (o 0) traccia una linea invisibile.

I valori predetti, o stimati, dal modello sono:

```
> valori.predetti <- alpha+beta*diametro
> valori.predetti
 [1]  5.103149  6.622906  7.636077 16.248033 17.261205 17.767790 18.780962
 [8] 18.780962 19.287547 19.794133 20.300718 20.807304 20.807304 22.327061
[15] 23.846818 28.406089 28.406089 30.432431 32.458774 32.965360 33.978531
[22] 34.991702 36.511459 44.110244 45.630001 50.695857 51.709028 53.735371
[29] 54.241956 54.241956 67.413183
```

Ovviamente, i valori predetti dal modello sono quelli che stanno sulla retta di regressione. Possiamo aggiungere questi punti nel grafico precedente con il comando

```
> points(diametro,valori.predetti,pch="X")
```

Il comando `points` aggiunge i punti in un plot esistente. L'opzione `pch` permette di scegliere il tipo di carattere da utilizzare nel grafico per identificare un punto (in questo caso si è scelto X).
I residui sono dati dalla differenza tra i valori osservati e quelli stimati

```
> residui <- volume - valori.predetti
> residui
 [1]  5.1968508  3.6770939  2.5639226  0.1519667  1.5387954  1.9322098
 [7] -3.1809615 -0.5809615  3.3124528  0.1058672  3.8992815  0.1926959
[13]  0.5926959 -1.0270610 -4.7468179 -6.2060887  5.3939113 -3.0324313
[19] -6.7587739 -8.0653595  0.5214692 -3.2917021 -0.2114590 -5.8102436
[25] -3.0300006  4.7041430  3.9909717  4.5646292 -2.7419565 -3.2419565
[31]  9.5868168
```

Il coefficiente di determinazione R^2 è dato da

```
> R2<- 1-var(residui)/var(volume)
> R2
[1] 0.9353199
```

Supponiamo ora che i dati in `volume` siano realizzazioni indipendenti di una variabile casuale con distribuzione normale con media $\alpha + \beta$`diametro` e varianza σ^2. Come ben noto, le stime di massima verosimiglianza di (α, β) coincidono con le stime dei minimi quadrati (ottenute in precedenza). La stima di massima verosimiglianza di σ^2 è data da:

```
> sigma2<- sum(residui^2)/n
> sigma2
[1] 16.91299
```

Possiamo calcolare la stima non distorta di σ^2

```
> s2<-sum(residui^2)/(n-2)
> s2
[1] 18.07940
```

che è equivalente a

```
> s2<-sigma2*n/(n-2)
> s2
[1] 18.07940
```

Utilizzando `s2` possiamo calcolare una stima della varianza di `alpha` e `beta`:

```
> var.alpha<-s2*(1/n+mean(diametro)^2/sum((diametro-mean(diametro))^2))
> var.alpha
[1] 11.3242
> var.beta<-s2/sum((diametro-mean(diametro))^2)
> var.beta
[1] 0.06119536
```

Un intervallo di confidenza di livello 0.95 per `alpha` avrà estremi inferiore e superiore, rispettivamente

```
alpha.lower<- alpha-qt(0.975,n-2)*sqrt(var.alpha)
alpha.upper<- alpha+qt(0.975,n-2)*sqrt(var.alpha)
> alpha.lower
[1] -43.82595
> alpha.upper
[1] -30.06096
```

dove abbiamo utilizzato la funzione `qt(p,df)`, che restituisce il quantile relativo alla probabilità `p` della distribuzione t di Student con `df` gradi di libertà (ricordate le tavole?). In modo analogo possiamo ottenere un intervallo di confidenza per `beta`

```
> beta.lower<- beta-qt(0.975,n-2)*sqrt(var.beta)
> beta.upper<- beta+qt(0.975,n-2)*sqrt(var.beta)
> beta.lower
[1] 4.559914
> beta.upper
[1] 5.571799
```

Verifichiamo ora l'ipotesi di nullità di `beta`, con alternativa bilaterale. Il valore osservato del test t_2 è

```
> test.t<-(beta-0)/sqrt(var.beta)
> test.t
[1] 20.47829
```

Sotto H_0, `test.t` ha distribuzione t di Student con $n-2$ gradi di libertà. Il valore di significatività osservato ($2\min\{Pr(T \geq t^{oss}), Pr(T < t^{oss})\}$) è quindi pari

```
> 2*min(pt(test.t,n-2),pt(test.t,n-2,lower.tail=F))
[1] 8.644334e-19
```

ed è equivalente a

```
> 2*pt(abs(test.t),n-2,lower.tail=F)
[1] 8.644334e-19
```

Abbiamo utilizzato la funzione `pt(q,df)`, che restituisce la probabilità relativa al quantile `q` della distribuzione t di Student con `df` gradi di libertà. L'opzione `lower.tail=F` indica che vogliamo la probabilità sulla coda destra, anziché su quella sinistra. Il valore ottenuto per il livello di significatività osservato è praticamente nullo, quindi l'ipotesi nulla è da rifiutare.
Se volevamo un test di livello fissato 0.05, dovevamo confrontare il valore di `test.t` con il quantile `qt(0.975,n-2)` che, in questo caso, è pari a 2.045230. Essendo `test.t` maggiore di 2.045230, l'ipotesi nulla veniva rifiutata (era comunque ovvio guardando il livello di significatività osservato).

Esercizio: verificare l'analoga ipotesi di nullità per `alpha`. Verificare inoltre l'ipotesi che `beta` sia uguale a 5.

Per finire, con il comando

```
> detach(Ciliegi)
```

diciamo a R che non stiamo più lavorando con il data frame `Ciliegi`. Ad esempio

```
> volume
Error: Object "volume" not found
> Ciliegi$volume
 [1] 10.3 10.3 10.2 16.4 18.8 19.7 15.6 18.2 22.6 19.9 24.2 21.0 21.4 21.3 19.1
[16] 22.2 33.8 27.4 25.7 24.9 34.5 31.7 36.3 38.3 42.6 55.4 55.7 58.3 51.5 51.0
[31] 77.0
```

Esercizio: ripetere tutte le operazioni fatte fin qui, utilizzando i logaritmi delle variabili `volume` e `diametro`.

Notiamo che, nell'esercizio precedente, abbiamo utilizzato la formula

$$\log(\texttt{volume}) = \alpha_1 + \beta_1 \log(\texttt{diametro})\,.$$

Questo, per le proprietà dei logaritmi, significa che

$$\log(\texttt{volume}) = \log(e^{\alpha_1}\texttt{diametro}^{\beta_1})\,.$$

e quindi che

$$\texttt{volume} = e^{\alpha_1}\texttt{diametro}^{\beta_1}\,.$$

Quindi, supponendo di aver salvato negli oggetti `alpha1` e `beta1` le stime dei coefficienti α_1 e β_1:

```
> beta1<-cov(log(volume),log(diametro))/var(log(diametro))
> alpha1<-mean(log(volume))-beta1*mean(log(diametro))
```

possiamo ottenere i valori predetti secondo questo modello nella scala originale delle variabili:

```
> valori.predetti1 <- exp(alpha1)*diametro^beta1
```

e poi confrontare i risultati graficamente con quelli precedenti:

```
> attach(Ciliegi)
> plot(diametro,volume)
> abline(alpha,beta,lty="dashed")
> lines(diametro,valori.predetti1)
> detach(Ciliegi)
```

La linea continua sembra adattarsi meglio alle osservazioni negli estremi. La funzione `lines` serve per aggiungere linee su un grafico esistente (vedi `help(lines)` per ulteriori dettagli).

Se definiamo i residui nella scala originaria delle variabili, come

```
> residui1<-volume-valori.predetti1
```

vediamo che la varianza di questi residui è minore rispetto a quella del primo modello utilizzato:

```
> var(residui1)
[1] 10.56006
> var(residui)
[1] 17.47675
```