Note prima lezione: R

Luigi Amedeo Bianchi

25 ottobre 2010

1 Introduzione

Il software open source R è disponibile al sito http://cran.r-project.org, dal quale si può non solo scaricare il programma, ma anche trovare pacchetti di comandi con cui estenderlo e guide per un approfondimento di certe tecniche.

In questa dispensa non graffieremo che la superficie di R, presenteremo solamente le prime tecniche per introdurne l'uso.

La prima cosa da conoscere di R sono i pacchetti. Le funzionalità matematiche di base stanno nel pacchetto `base`, quelle statistiche in `stats` e quelle grafiche in `graphics`. Per il momento questi sono i soli pacchetti che ci interessano: non li useremo nemmeno al massimo del loro potenziale.

Ogni operazione viene eseguita in R in seguito ad opportuni comandi. Per comodità conviene tenere aperta, accanto alla console di R, una finestra di un editor di testo, con un file di testo semplice (`.txt`) aperto, in modo da poter scrivere tutti i comandi che ci servono e averli tutti ben ordinati, per poi salvarli e riutilizzarli nel futuro. Una volta scritti uno o più comandi nel file possiamo copiarli e incollarli nella console per eseguirli. Ci sono anche metodi più sofisticati per fare ciò, ma per il momento possiamo accontentarci di questo semplice approccio.

Quando si inizia ad utilizzare un software la cosa più importante è l'`help`. In R questo può essere invocato in almeno due modi differenti. Il primo è molto classico: dal menu `help` o `aiuto` selezionate la voce `R Help` o `cerca nella guida`. La guida è in lingua inglese, ma questo non dovrebbe essere un grosso ostacolo. Il secondo metodo, molto comodo se ci ricordiamo il nome di un comando, ma non la sua sintassi precisa, è digitare `help(nomecomando)` nella console di R. Una delle parti più utili dell'help per ciascun comando è, ove presente, la sezione Examples, in fondo, che ci dà alcuni esempi di utilizzo del comando stesso.

2 Primi esempi di codice R

Cominciamo a vedere alcuni semplici esempi di uso del software R. Cominciamo con il tracciare i grafici di alcune distribuzioni di probabilità.

La distribuzione che sicuramente conoscete meglio è la Gaussiana, quindi la lasciamo come esercizio. Consideriamo invece la distribuzione Beta, che è una distribuzione (o meglio una famiglia di distribuzioni) a due parametri. Per prima cosa, dal momento che stiamo lavorando su un computer, dobbiamo discretizzare: vogliamo quindi prendere un certo numero di valori per x nell'intervallo $(0, 1)$ su cui la distribuzione è definita. Per fare ciò (attenzione che dobbiamo escludere gli estremi dell'intervallo dai nostri punti) abbiamo diverse strategie possibili. Il metodo standard è il seguente: `x=seq(0.001,0.999,by=0.001)`. In alternativa possiamo anche digitare `x <- (1:999)/1000` e ottenere lo stesso risultato. Osserviamo che l'assegnazione si può fare sia con `=` che con `<-`. Con questi comandi abbiamo creato un vettore di 9999 valori equispaziati tra 0 e 1. Se ora digitiamo `x` nella console di R otterremo una stampata a schermo dei valori delle componenti di x.

Ora dobbiamo creare il corrispondente vettore delle densità. Per fare ciò usiamo l'help. Se digitiamo `help(beta)` otteniamo la pagina che riguarda la *funzione* beta, che non è quello che ci interessa. Proviamo allora con `?? 'beta distribution'` che ci dà tutte le pagine inerenti l'argomento cercato (almeno secondo il computer). Abbiamo allora un elenco di pagine che ci potrebbero interessare, tra cui la pagina `Beta` relativa al pacchetto `stats`. Questa è proprio quella che ci interessa. In generale le distribuzioni statistiche saranno contemplate nel pacchetto `stats`.

Leggiamo allora la pagina di help e vediamo che per avere la densità di una Beta di parametri, diciamo, $\alpha = 1.5$ e $\beta = 2.3$ il comando che dobbiamo dare è

```
y = dbeta(x,1.5,2.3)
```

che genera il vettore delle densità della stessa lunghezza del vettore `x`. Ora vogliamo darne una rappresentazione grafica (anche perché leggere i valori del vettore `y` non è proprio intuitivo). Per fare questo il comando è `plot(x,y)`. Se osserviamo l'output notiamo che non è tanto bello. Per fortuna il comando `plot` si potra dietro un buon numero di parametri opzionali, per migliorare l'aspetto del nostro grafico. Ne vediamo solo alcuni (per gli altri vi rimando all'help). Digitiamo dunque, sulla console di R, `plot(x,y,type='l',col='blue')`: otterremo un grafico di tipo linea continua (`type='l'`) e di colore blu (`col='blue'`). Meglio, vero?

Ora (senza chiudere l'immagine ottenuta) vogliamo vedere anche come è fatta la corrispondente funzione di ripartizione. Sempre dalla pagina di help relativa alla distribuzione Beta scopriamo che il comando per fare questo è `z = pbeta(x,1.5,2.3)`. Possiamo stampare il grafico con `plot(x,z)` come prima, ma può anche essere interessante stampare sulla stessa finestra grafica, sovrapposti, i due grafici, della densità e della funzione di ripartizione. Per fare questo, dopo aver lanciato `plot(x,y,type='l',col='blue')` digitiamo nella console (senza chiudere la finestra del grafico) `par(new=TRUE)`

e ancora `plot(x,z,type='l',col='red')`. L'immagine che ne risulta può essere migliorata ulteriormente, usando in modo opportuno i parametri di `plot`.

Esercizio 1. Come esercizio provate a fare queste cose per la distribuzione gaussiana (o normale), scegliendo in modo opportuno i valori da assegnare a `x`. Inoltre potete provare a fare lo stesso per qualcuna delle altre distribuzioni che trovate nel pacchetto `stats`.

Più interessante è confrontare due realizzazioni di una passeggiata aleatoria (random walk). Cominciamo con la prima.

Innanzitutto generiamo 10000 numeri pseudocasuali[1] estraendoli da una distribuzione normale standard:

```
delta.x=rnorm(10000,0,1)
```

essi rappresentano i passi relativi ad ogni istante. Consideriamo il vettore posizione, anch'esso di lunghezza 10000. Dal momento che i valori vengono inizializzati all'interno di un ciclo `for`, dobbiamo dichiarare prima la lunghezza del vettore. Un modo facile per fare ciò è

```
x=delta.x
```

sfruttando il fatto che i valori di `x` verranno comunque riaggiornati e che `delta.x` ha lunghezza 10000. Allora il ciclo `for` sarà

```
x[1]=delta.x[1]
for (i in 2:10000){
x[k] = x[k-1]+delta.x[k]
}
plot(x.type='l')
```

Questa è una possibile realizzazione. Senza chiudere la finestra grafica torniamo alla console e proviamo a realizzare nuovamente una passeggiata.

```
delta.x=rnorm(10000,0,1)
x[1]=delta.x[1]
for (i in 2:10000){
x[k] = x[k-1]+delta.x[k]
}
```

Per rappresentarla graficamente sovrapposta all'altra immagine usiamo il comando `lines`:

```
lines(x,col='red')
```

Esercizio 2. Tracciare il grafico di due realizzazioni di una passeggiata aleatoria con passo distribuito secondo un'altra distribuzione.

[1] I numeri generati da un computer non sono mai davvero casuali

3 Caricare dati

Possiamo caricare dati in svariati modi. Vediamone rapidamente alcuni.

Se abbiamo pochi dati la cosa più conveniente è inserirli manualmente in R con il comando `x=c(1.1,2.5,6,2.3,400)`. Tuttavia spesso abbiamo molti dati, magari già salvati in una riga di un file Excel. In questo caso possiamo procedere in questo modo: selezioniamo i dati che ci interessano e li copiamo. Passiamo nella console di R e digitiamo `X=scan(clipboard,dec=',')`. In questo modo assegnamo al vettore `x` i valori presenti nella riga in Excel. Osserviamo il parametro `dec=','`: esso dice a R che i dati in Excel sono scritti secondo il formato italiano, in particolare che la virgola è il separatore delle cifre decimali (contrapposta al punto usato nel mondo anglosassone e in R).

Vedremo poi altri metodi più sofisticati per recuperare dati da tabelle.