

Breve introduzione a R. (Maurizio Pratelli)

Queste brevissime note sono una presentazione sintetica di quella piccola porzione del software **R** che è stata presentata nel corso, e si appoggiano molto per gli esempi alle esercitazioni che sono state svolte e che si trovano nella mia Home Page. Queste note non possono essere portate alla prova scritta (dove si può portare solo il “Kit per l’esame”).

Come per tutti i software, **R** non può essere appreso solo leggendo queste o simili note: bisogna aprire il programma e provare, provare, provare...

Per prima cosa bisogna ovviamente procurarsi il programma **R** : questo può essere fatto collegandosi al sito <https://www.r-project.org/> , il programma è libero e ci sono versioni per *Windows, Linux e Apple-OS* con piccole differenze tra i pannelli che si aprono. Non mi risulta che ci sia una versione ufficiale per Android ma in rete si trovano alcuni espedienti per riuscire a caricarlo anche su tablet e smartphone (però è molto poco agevole usare un software statistico su uno smartphone...)

E’ conveniente caricare anche *RStudio* : questo fornisce un editor più pratico e facilita alcune operazioni di ricerca e salvataggio di fogli di lavoro e soprattutto appare identico sotto *Windows, Linux e OS*; *RStudio* deve essere caricato *dopo* aver già caricato R.

Quando inizia una **sessione di lavoro** con **R** il programma automaticamente ricarica l’ultima sessione di lavoro salvata, e quando la sessione termina il programma chiede se si vuole che sia memorizzata. Si può salvare (e poi caricare) un foglio di lavoro con un nome diverso, l’estensione del file è comunque sempre **.Rdata**.

All’inizio di una riga di comando compare il simbolo **>** (questo simbolo non deve essere digitato, compare automaticamente). Per avere l’elenco degli oggetti presenti nel foglio di lavoro si deve digitare **> ls()** , per rimuovere un oggetto (chiamato poniamo “nome”) si digita **> rm(nome)** e per rimuovere l’intera lista si digita **> rm(list=ls())**

I **comandi** possono essere dati, equivalentemente, usando i simboli “<-” oppure “=” : occorre però precisare che il simbolo “=” va inteso da destra a sinistra. Ad esempio se si scrive **> x=3+5** viene assegnato il numero 8 al simbolo **x**, ma se si scrive **> 3+5=x** il programma dà un messaggio di errore.

Per ottenere informazioni su un comando (ad esempio **sum** che verrà introdotto tra poco) si può digitare **> ?sum** .

Il primo uso di **R** può essere semplicemente come calcolatrice: le operazioni di aritmetica sono indicate dai simboli **+** **-** ***** **/** e la potenza si indica con l’accento circonflesso oppure col doppio asterisco (si può scrivere **a^b** o equivalentemente **a**b**).

Le principali **funzioni matematiche** sono **sqrt(.)** **exp(.)** **log(.)** per indicare radici ed esponenziali, le funzioni trigonometriche sono **sin(.)** **cos(.)** **tan(.)** **acos(.)** **asin(.)** **atan(.)** **sinh(.)** **cosh(.)** **tanh(.)** mentre il valore assoluto è indicato **abs(.)**. Per *arrotondare* il numero **x** alla **h**-ma cifra decimale si usa **round(x,h)**, ed il numero **p**-greco è **pi** mentre non c’è un simbolo per il numero **e**. Per ottenere **e** si può scrivere **exp(1)**; attenzione si deve scrivere (per 2 **p**-greco) **2*pi** e non **2pi** (comparirebbe un messaggio d’errore).

I principali oggetti su cui lavora **R** sono i *vettori*, le *matrici* ed i *data-frames*. I singoli numeri sono considerati vettori di lunghezza 1. I nomi assegnati a vettori o matrici possono essere qualsiasi, contenere dei punti *ma non delle virgole* (le virgole sono sempre separazione tra due

oggetti diversi, i decimali si indicano col punto). Ci concentreremo soprattutto sui vettori che sono l'oggetto più frequente.

Classi di oggetti: gli elementi dei vettori, matrici ecc... appartengono a *classi* differenti, le principali classi di cui faremo uso sono "integer" (interi), "numeric" (numeri), "logical" (caratteri logici) e "character". Se x è un generico vettore (o matrice..) per conoscere a quale classe appartiene si può digitare `class(x)`.

Mentre su interi e numeri non ci sono dubbi, rinviamo a dopo l'analisi sui caratteri logici ed insistiamo su "character": sono caratteri un insieme di simboli, che può contenere anche numeri e che viene definito compreso tra virgolette. Se ad esempio voglio definire un elemento x corrispondente proprio ad un character (ad esempio "prova") dovrò digitare `> x="prova"`.

Inserimento di dati: per costruire un vettore x che ha come componenti ad esempio a , b e c si può usare il comando `> x=c(a,b,c)` (`c(. , .)` sta per *concatenazione*). Questo comando può essere usato anche su vettori di lunghezze diverse ad esempio

```
> x=c(3,5,4)
```

```
> y=c(7,x)
```

Il comando `x=c(a,b,c)` va bene se a,b,c sono dei numeri (o sono stati predefiniti, come ad esempio `a=7 ...`), ma se voglio definire un vettore i cui elementi sono i caratteri a, b, c devo scrivere `x=c("a","b","c")`.

Il comando `seq(a,b,by=c)` o semplicemente `seq(a,b,c)` indica una sequenza di numeri da a a b con passo (cioè distanza tra due numeri) eguale a c : provare ad esempio il comando `seq(2.3,4.5,0.4)` (in luogo di `0.4` si può scrivere semplicemente `.4`).

`seq(a,b)` equivale a `seq(a,b,1)` e, se a e b sono interi, `a:b` equivale a `seq(a,b)`. Il comando `rep(a,n)` (dove n è un intero positivo) costruisce un vettore di lunghezza n le cui componenti sono tutte eguali ad a .

Per **estrarre dati** da un vettore (o più avanti da una matrice) si usano le parentesi quadre (è l'unico caso in cui compaiono queste parentesi): ad esempio `x[i]` estrae la componente i -ma di x e `x[i:j]` estrae le componenti dalla i -ma alla j -ma. All'interno delle parentesi quadre si possono mettere condizioni anche più complesse: supponiamo ad esempio di avere un vettore di numeri X (di lunghezza sconosciuta) e di voler definire un nuovo vettore Y formato da tutte le componenti di posizione *dispari* (cioè il primo elemento, il terzo e così via...): questo si può fare ad esempio col comando `> Y=X[seq(1,length(X),2)]`. Il comando `length()` è illustrato una riga sotto.

Più avanti vedremo che si possono estrarre dati anche con comandi logici.

Operazioni sui vettori: si possono usare i comandi `x+y` e `x*y` a condizione che la lunghezza di un vettore sia un multiplo della lunghezza dell'altro, ci sono poi i comandi `length(x)` (lunghezza), `min(x)`, `max(x)`, `range(x)` (minimo e massimo, `range()` fornisce sia il minimo che il massimo), `sort(x)` (mette in ordine crescente gli elementi di x), `sum(x)`, `cumsum(x)` (cioè somma cumulata), `prod(x)` (per capire questi comandi bisogna provarli su vettori concreti).

Ad esempio per ottenere $n!$ si può scrivere `prod(1:n)`.

Non esiste un comando per mettere in ordine *decrescente* gli elementi di x , ma questo si può ottenere facilmente ad esempio col comando `-sort(-x)`.

Ci sono poi i **comandi statistici** `mean(x)`, `var(x)`, `sd(x)` (media, varianza e deviazione standard campionarie), `quantile(x,p)` (il p -quantile) e `summary(x)`. Il comando `summary` riporta minimo, primo quartile, media e mediana, terzo quartile e massimo del vettore x .

Se x e y due vettori di dati con la stessa lunghezza, si possono dare i comandi `cov(x,y)` e `cor(x,y)` (covarianza campionaria e coefficiente di correlazione).

Matrici. Supponiamo di voler costruire una matrice con 5 righe e 2 colonne i cui elementi sono 10 numeri che inseriamo in un vettore di nome x (naturalmente di lunghezza 10): il comando $A=matrix(x,5,2)$ organizza il vettore x in una matrice 5 righe e 2 colonne, ed è equivalente a $A=x$ $dim(A)=c(5,2)$. Per estrarre dalla matrice A l'elemento di posizione i,j si scrive $A[i,j]$, mentre $A[i,]$ estrae la i -ma riga e $A[,j]$ la j -ma colonna.

Data.frames : i *data.frames* sono liste di vettori, anche di classi diverse, ma della stessa lunghezza (le matrici contengono esclusivamente elementi della stessa classe). Se X,Y,Z hanno la stessa lunghezza, $data.frame(X,Y,Z)$ li unisce in un *data.frame* del quale X,Y,Z formano le colonne. Le colonne dei *data.frames* hanno sempre un nome che può essere fornito (ad esempio col comando precedente i nomi sono proprio X,Y e Z) oppure di *default* è eguale a $V1,V2$ ecc..

Questo succede ad esempio quando un *data.frame* è inserito da una tabella, come vedremo più avanti.

Per estrarre da un *data.frame* A la i -ma colonna (poniamo di nome V) si può agire come con le matrici $X=A[,i]$ oppure scrivendo $X=A$V$.

In conclusione i *data.frames* hanno le proprietà delle matrici ma sono molto più "aperti".

Inserimento dati da tabelle o testi: supponiamo di avere una tabella di dati (numerici o anche "caratteri") contenuta ad esempio in un foglio Excel o in un file .pdf.

Per importare i dati sotto forma di vettore, dopo aver marcato inizio e fine dei dati da importare (eventualmente col comando `ctrl c`) si scrive `scan("clipboard")`. Se questi dati sono numerici ed il decimale è indicato con la virgola, occorre aggiungere `dec=","` (altrimenti vengono interpretati come caratteri).

Viceversa per importare i dati sotto forma di *data.frame* il comando è `read.table("clipboard")` (sempre eventualmente con l'aggiunta di `dec=","`).

Con Apple i comandi sono leggermente diversi: `scan(pipe("pbpaste"))` e `read.table(pipe("pbpaste"))`.

Le variabili booleane sono soltanto due: `TRUE` e `FALSE` (che possono anche essere abbreviate con `T` ed `F`, a meno che `T` ed `F` non siano già usate per denominare altri oggetti). Naturalmente `T` ed `F` non sono dei numeri, però convenzionalmente a `T` si attribuisce il valore 1 e ad `F` il valore 0 (provare a scrivere ad esempio `T+7` oppure `F*5`). Il motivo di questa attribuzione sarà chiaro qualche riga sotto.

Gli operatori logici sono : `<` `<=` `>` `>=` `==` `!=` (diverso) (attenzione: il simbolo `=` non è un operatore logico).

Ci sono poi i **connettivi logici** `&` ("e", congiunzione) `|` ("o", disgiunzione) `!` ("non", negazione). Questi connettivi logici possono anche essere usati (all'interno delle parentesi quadre) per isolare parti di un vettore, di una matrice o di un *data.frame*.

Supponiamo ad esempio che X sia un vettore di dati numerici: se si dà il comando $X>=0$ compare una sequenza di `TRUE` e `FALSE` a seconda che il termine corrispondente sia positivo oppure no, se si scrive `sum(X>=0)` si ottiene il numero di elementi di X che sono positivi e per ottenere la percentuale di termini positivi si può scrivere `sum(X>=0)/length(X)`. Scrivendo invece $X[X>=0]$ si ottiene un nuovo vettore formato dalle componenti di X che sono positive, se voglio isolare le componenti $X[i]$ tali che $1<=X[i]<=5$, scriverò $X[X>=1 \& X<=5]$ e così via ... (i comandi possono anche riguardare colonne di un *data.frame* diverse, vedere

ad esempio l'analisi del Questionario citato sopra...)

Comandi grafici di primo livello (**hist**, **plot**, **curve**)

I comandi grafici di primo livello aprono una finestra e vi tracciano qualcosa, ad esempio se x è un vettore di dati (numerici) il comando `hist(x)` ne traccia l'istogramma a bastoni. Il numero di bastoni o classi è di default 10 e se i dati sono numerosi è opportuno ampliare il numero di classi scrivendo `hist(x,h)` dove h è il numero di classi desiderate (non bisogna esagerare, se la lunghezza di x è ad esempio 250 ha senso scrivere `hist(x,30)` ma non `hist(x,120)`).

Con questi comandi la frequenza dei dati è assoluta, per ottenere viceversa la frequenza relativa occorre scrivere `hist(x,h,freq=FALSE)` o più brevemente `hist(x,h,freq=F)`. Ad esempio se si vuole sovrapporre a un istogramma una densità è *essenziale* inserire nel comando `freq=FALSE` altrimenti la densità verrebbe *schacciata*.

Si può poi aggiungere `xlab=".."` `ylab=".."` `main=".."` per dare un titolo all'asse x , all'asse y o all'intero istogramma.

Se x e y sono due vettori di eguale lunghezza, il comando `plot(x,y)` traccia i punti di coordinate $x[i], y[i]$; si può aggiungere al comando `type="p"` (punti), `type="l"` (linee), `type="s"` (step, cioè gradini), e anche `type="n"` (nulla).

Il comando `type="n"` sembra insensato, ma ci sono delle situazioni nelle quali invece è utile (vedi esercitazione del 12 aprile, ultima parte).

Anche nel comando `plot` qui si può aggiungere `xlab`, `ylab`, .. e si può colorare col comando `col="red"` ("green", "blue", "yellow" ...)

Infine il comando `curve(f(x),a,b)` traccia il grafico della funzione f tra a e b ; inoltre si deve scrivere proprio x (o eventualmente, ma non conviene, un altro carattere ma poi aggiungere `xname="."`, ad esempio `curve(sin(t),0,2*pi,xname="t")`).

Per tracciare il grafico, poniamo della funzione $f(x)=x+x^2$ tra -1 e 2, si può procedere in due modi che danno lo stesso risultato:

```
> curve(x+x^2, -1, 2)
```

oppure

```
> x=seq(-1,2,.02)
> y=x+x^2
> plot(x,y,type="l")
```

Il primo modo è decisamente più rapido, ma non sempre si può usare.

Nel tracciare punti o curve, il comando `lwd=2` (ma anche `lwd=3` ...) traccia linee più spesse, in un `plot` il comando `pch=...` inserisce simboli diversi (ad esempio `pch=19` traccia dei punti più spessi), e naturalmente si può colorare

Per rappresentare due grafici (o istogrammi) in due finestre separate sovrapposte si dà il comando `par(mfrow=c(2,1))` invece in due finestre affiancate `par(mfrow=c(1,2))` (si possono dare anche numeri diversi, ad esempio `par(mfrow=c(2,3))`); è importante sottolineare che il comando `par(mfrow=c(...))` deve essere dato *prima* di tracciare istogrammi o grafici.

Comandi grafici di secondo livello

I comandi grafici di secondo livello sovrappongono, in una finestra già esistente, un nuovo

grafico o un plot ...

Il comando **lines** sovrappone un "plot" di tipo "l" a un istogramma o a un plot esistente (il comando è **lines(x,y)**, è superfluo scrivere **type="l"**, è sbagliato scrivere **type="p"**). Naturalmente si può inserire **lwd= ...** oppure **colorare**.

In modo analogo il comando **points** sovrappone un "plot" di tipo "p".

Per sovrapporre un grafico ad un altro, o a un istogramma, già esistente, si può aggiungere, all'interno del comando **curve** l'istruzione **add=TRUE** (o anche più brevemente **add=T**).

Infine il comando **abline(A,B,...)** (eventualmente con l'aggiunta di colore e **lwd**) traccia la retta di intercetta A e coefficiente angolare B (insomma la retta $y=A+Bx$).

I comandi qqplot, qqnorm e qqline: il comando **qqplot(X,Y)** confronta i quantili di due vettori di dati X e Y aventi eguale lunghezza, il comando **qqnorm(X)** confronta i quantili di X con quelli della gaussiana standard. Se gli elementi di X hanno una distribuzione ragionevolmente gaussiana, i punti che vengono disegnati dal comando **qqnorm** tendono ad essere disposti su una retta: questo confronto è facilitato dal comando **qqline(X)** che sovrappone la retta passante per il primo ed il terzo quartile.

Comandi sulle variabili aleatorie: partiamo dalla variabile più usata, cioè la variabile gaussiana. I comandi **dnorm(x)** **pnorm(x)** **qnorm(p)** **rnorm(n)** sono rispettivamente la densità in x, la funzione di ripartizione (c.d.f) in x, il p-quantile e la generazione di n numeri casuali per una variabile gaussiana $N(0,1)$; per variabile gaussiana con media m e scarto quadratico medio s si usa **dnorm(x,m,s)**, **pnorm(x,m,s)** ...

Le lettere d, p, q, n ... si ripetono per tutte le variabili indicando sempre la densità, la funzione di ripartizione, il quantile e la generazione casuale ... naturalmente se la variabile è *discreta*, con d si intende la *densità discreta* o *funzione di massa*.

dbinom(h,n,p) è la densità in h di una v.a. Binomiale di parametri n e p e analogamente per **pbinom(..)**, **qbinom(..)**, **rbinom(..)**.

dpois(h,lambda) per una variabile di Poisson di parametro lambda

dunif(x,a,b) per una variabile con densità uniforme su [a,b] (se non si riportano a e b, **dunif(x)=dunif(x,0,1)**)

dexp(x,lambda) per una variabile esponenziale di parametro lambda (se non si riporta lambda, **dexp(x)=dexp(x,1)**)

dchisq(x,n) per una variabile chi-quadro a n gradi di libertà

dt(x,n) per una variabile di Student a n gradi di libertà

dgamma(x,a,lambda) per una variabile Gamma con "shape" a e "rate" lambda
(**dgamma(x,a)=dgamma(x,a,1)**)

dweibull(x,a,s) per una variabile di Weibull con "shape" a e "scale" s
(**dweibull(x,a)=dweibull(x,a,1)**)

Definizione di funzioni.

Queste cose si imparano bene su esempi concreti, il primo esempio che abbiamo visto è stato il comando per tracciare la funzione di ripartizione empirica (esempio visto il giorno 8 marzo). Riporto i comandi:

```
emp.cdf=function(X)
{
# traccia la funz. ripartizione empirica
# X vettore di dati
Y=sort(X)
Z=seq(1,length(X))/length(X)
plot(Y,Z,type="s")
}
```

Come si vede, dopo aver dato un nome alla funzione, si apre una parentesi { e si danno, su righe diverse, le istruzioni e si conclude chiudendo la parentesi }. Le righe che cominciano con # vengono ignorate (servono come promemoria).

Ad esempio, vogliamo generare 20 numeri a caso tra 0 e 2 e tracciarne la funzione di ripartizione empirica: dopo aver definito `emp.cdf` si può procedere così:

```
> X=runif(20,0,2)
> emp.cdf(X)
ma anche più brevemente
> emp.cdf(runif(20,0,2))
```

Comandi if .. else e cicli for e while.

Anche questi si imparano su esempi concreti: `if (. .) { .. } else { . . }` dentro la parentesi (. .) si inserisce la condizione e dentro { .. } il comando conseguente.

Quando c'è una sequenza di operazioni ripetute, si può usare il ciclo `for`: il comando è `for (i in 1:n) { ... }`

E' sostanzialmente equivalente il comando `while`, che però ha un uso un po' diverso:
`i=1 while(i<=n) { {...} i=i+1}`

Faccio riferimento alla seconda parte dell'esercitazione del 12 aprile, interamente dedicata a questi comandi.

Esempi finali.

La teoria ci dice che sommando i quadrati di 4 variabili gaussiane standard si ottiene una variabile chi-quadro a 4 gradi di libertà, cioè una Gamma di parametri 2 e 1/2: vogliamo verificarlo sperimentalmente generando 4 campioni di numerosità 1000 con distribuzione gaussiana standard e sovrapponendo all'istogramma della somma dei quadrati la densità Gamma. Si può fare con i comandi

```
> X1=rnorm(1000)
> X2=rnorm(1000)
> X3=rnorm(1000)
> X4=rnorm(1000)
> Y=X1^2+X2^2+X3^2+X4^2
> hist(Y,50,freq=F)
> curve(dgamma(x,2,.5),add=T,col="red")
```

Si può constatare che la densità si adatta molto bene all'istogramma, ribadiamo che è *essenziale* l'istruzione `freq=F`

Terminiamo con un altro esempio che in realtà è più un quiz: sappiamo che se X è gaussiana standard, C chi-quadro a n gradi di libertà indipendente, posto

$$T = \sqrt{n} * X / \sqrt{C} \quad (\text{le notazioni sono quelle di } \mathbf{R})$$

si ottiene una variabile di Student a n gradi di libertà. Di nuovo vogliamo verificarlo sperimentalmente (per $n=4$) con i comandi

```
> X=rnorm(1000)
> C=rchisq(1000,4)
> T=2*X/sqrt(C)
> hist(T,50,freq=F)
> curve(dt(x,4),add=T,col="red")
```

Ma questa volta le cose non funzionano (l'istogramma viene cancellato), infatti è stato commesso un errore: quale?