Breve Introduzione a R

Guida rapida per iniziare ad usare l'ambiente di calcolo R

M. Ponti
Centro Interdipartimentale di Ricerca per le Scienze Ambientali in Ravenna, Università di Bologna

Aprile, 2003

Prefazione

Queste brevi note sono rivolte a quanti possiedono buone conoscenze di statistica ed in particolare di Analisi della Varianza e che vogliono avvicinarsi all'uso di R pur non avendo esperienze di programmazione.
Senza pretese di approfondire la programmazione in R, né tanto meno di affrontare complessi concetti di statistica, questa guida vuole essere d'aiuto nel muovere i primi passi in questo ambiente di lavoro ottenendo subito dei risultati tangibili. L'utente, dopo aver superato il primo impatto con l'interfaccia decisamente poco "amichevole", potrà rendersi conto delle enormi potenzialità di R nell'elaborazione dei dati e nella presentazione grafica dei risultati.

Ringraziamenti

Il primo ringraziamento va alla Prof.ssa Marti Jane Anderson, docente e ricercatrice presso il Dipartimento di Statistica dell'Università di Auckland (Nuova Zelanda). Grazie a lei ho mosso i miei primi passi in R durante il corso "*Design and analysis of monitoring and experiments for environmental scientist*" organizzato dall'Università di Bologna nel marzo 2003. Gran parte delle seguenti note derivano dagli appunti del corso, oltre che dalle guide fornite con R. Ringrazio inoltre i miei colleghi del Gruppo di Ecologia dell'Università di Bologna che hanno contribuito alla revisione di queste note.

Sommario

1 Introduzione a R

R è un ambiente d'elaborazione che consente di manipolare dati, eseguire calcoli e procedure complesse nonché rappresentare graficamente i risultati. Possiede numerose funzioni dedicate al calcolo matematico avanzato (calcolo matriciale, operazioni con numeri complessi, ecc.) e all'analisi statistica. Consente di realizzarne, con un minimo di sforzo nella programmazione, nuove funzioni e procedure da richiamare all'occorrenza. Fra i suoi numerosi pregi vi è quello di trattare dati, risultati e programmi come oggetti facilmente richiamabili e manipolabili.

R è stato inizialmente creato da Robert Gentleman e da Ross Ihaka del Dipartimento di Statistica dell'Università di Auckland (Ihaka e Gentleman, 1996). Tra il 1997 e il 1999 si sono aggiunti 16 ricercatori costituendo così il principale gruppo di sviluppo. Oggi sono molti i ricercatori che contribuiscono fornendo aggiornamenti ed implementazioni. La filosofia di R è quella di un ambiente di sviluppo "aperto", di libero utilizzo e dove chiunque può contribuire e condividere le proprie esperienze e applicazioni. R è un programma di libera diffusione disponibile per tutti i principali sistemi operativi (Unix, Windows, Mac OS) attraverso il sito internet http://www.r-project.org/.

Insieme al software si possono scaricare dal sito principale e dai numerosi siti collegati sia i manuali d'uso sia numerosi pacchetti aggiuntivi. In particolare si raccomanda fin da subito la lettura della Guida Introduttiva (Venables e Smith, 2002) che fornisce i rudimenti di R e da cui è liberamente tratta la prima parte di queste note in italiano.

R è sostanzialmente, come primo approccio, un interprete di comandi che esegue passo dopo passo le istruzioni che gli vengono impartite. Può essere utilizzato in modo interattivo, impartendo un comando ed ottenendo un risultato. Conserva in memoria i dati e i risultati e fornisce un output solo quando richiesto. All'avvio appare semplicemente come una "console" con un cursore, rappresentato dal simbolo '>', dove inserire i comandi e le istruzioni. Nulla di più! Il cursore resta immobile ed attende i vostri comandi: nulla verrà eseguito fino a quando non sarà richiesto e nulla di ciò che verrà elaborato sarà visualizzato se non specificato. Non lasciatevi spaventare da quest'aspetto apparentemente spartano: dietro si cela un motore matematico e grafico potentissimo!

1.1 I primi passi

Non è certo scopo di questa guida quello di apprendere tutte le potenzialità di R né tanto meno di spiegarne nel dettaglio tutte le funzioni e comandi. Quindi di seguito saranno descritti solo i comandi di base essenziali per muovere i primi passi. Tra i pochi comandi disponibili nella barra dei menu sono d'immediato utilizzo i seguenti:

Edit > Clear console (Ctrl + L)

Cancella dalla console i testi presenti ma non rimuove nessun dato o comando dalla memoria.

Misc > List objects

Equivale al comando `ls()` e fornisce la lista degli "oggetti" in memoria. Per oggetti si intendono le variabili e le funzioni memorizzare, se la memoria è vuota compare la scritta `character(0)`.

Misc > Remove all objects

Equivale al comando `rm(list=ls(all=TRUE))` e cancella dalla memoria tutti gli "oggetti" presenti.

NB: prima di incominciare una nuova analisi o elaborazione in R è importante essere certi che non vi siano oggetti con lo stesso nome che possano sovrapporsi, è quindi consigliabile ogni volta rimuovere tutti gli oggetti e ricominciare da capo, almeno fino a quando non si avrà una sufficiente familiarità con l'ambiente di lavoro.

Per impartire un'istruzione occorre scriverla correttamente e dare l'invio. Una soluzione pratica è quella di scrivere tutte le funzioni di uso comune in un file di testo per poterle all'occorrenza copiare e incollare direttamente nella console. È possibile selezionare col mouse i testi presenti nella console, copiarli ed incollarli in un file di testo. È anche possibile copiare le istruzioni dalla versione elettronica di questa guida ed incollarle

direttamente nella console. Per questo motivo saranno di seguito sempre riportati anche tutti i comandi preliminari in modo da consentire un'immediata esecuzione da parte dell'utente.

Prima di cominciare è però importante sapere che:

- R è "*case sensitive*" cioè considera come caratteri differenti le lettere maiuscole e minuscole, quindi `A` e `a` sono ad esempio nomi di variabili diverse
- tutte le lettere ed i numeri possono essere utilizzate per dare nomi a variabili, funzioni e procedure; sono però esclusi gli spazi e tutti i segni di punteggiatura ad eccezione del punto ('.'), comunemente usato per separare le parole che compongono il nome; ogni nome deve cominciare con una lettera e non con un numero
- ogni comando deve essere separato da un punto e virgola (';') oppure deve essere su una nuova linea
- testi di commento possono essere aggiunti dopo il simbolo cancelletto ('#'), tutto ciò che si trova dopo questo simbolo e fino alla riga successiva viene ignorato da R
- se si va a capo prima della conclusione di un comando R mostrerà all'inizio della riga il simbolo più ('+')
- è possibile richiamare i comandi impartiti in precedenza utilizzando la freccia 'su' della tastiera

Convenzioni adottate:
I comandi di R, immediatamente copiabili ed eseguibili, sono scritti con il carattere `Courier New`. I nomi delle variabili e degli oggetti, scritti in verde, possono essere cambiati a piacimento dall'utente a patto che il nome utilizzato rimanga coerente in tutti i comandi che seguono. Gli esempi di *output* di R sono scritti in blu.

1.2 I primi comandi

I primi comandi da imparare sono:

```
ls()
```

Fornisce la lista dei nomi degli oggetti in memoria.

```
rm(lista.oggetti)
```

Rimuove tutti gli oggetti nominati nell'elenco tra parentesi, i nomi vanno separati da virgole. Un esempio particolare è dato dal comando seguente:

```
rm(list=ls(all=TRUE))
```

In questo caso viene rimossa la lista di oggetti `list` ottenuta col comando `ls(all=TRUE)` che consente di includere non solo gli oggetti ma anche l'archivio dei comandi digitati in precedenza.

```
help(nome.funzione)
```

oppure

```
?(nome.funzione)
```

Richiama in una nuova finestra le spiegazioni associate alla funzione o al comando specificati. Le spiegazioni sono disponibili per tutti i comandi e le funzioni fornite di serie con R.

Molti altri comandi saranno introdotti di volta in volta e spiegati al momento.

1.3 Tipi di dati

R è in grado di gestire con la stessa disinvoltura **numeri reali**, **numeri complessi**, **stringe di testo**, e **valori logici**. I numeri complessi si distinguono per la presenza della parte immaginaria, che deve sempre essere esplicitata, ad esempio:

```
sqrt(-17+0i)
```

Le stringhe, cioè insiemi di caratteri, sono racchiuse da doppie virgolette.

```
"testo"
```

I valori logici disponibili sono `TRUE`, `FALSE` e `NA` che indica "risposta non disponibile". I primi due valori possono essere abbreviati rispettivamente con `T` e `F` (ci si ricordi di usare sempre le lettere maiuscole!).

1.4 Variabili e vettori

L'utente può creare una variabile, o qualunque "oggetto" desideri, nel momento in cui gli serve, senza la necessità di una dichiarazione esplicita iniziale. Se da una parte questo garantisce grande flessibilità, dall'altra obbliga a porre molta attenzione a ciò che si crea e si maneggia. In qualunque momento è possibile richiamare la lista degli oggetti e visualizzarne il contenuto semplicemente digitandone il nome. L'assegnazione di un valore ad una variabile può avvenire in diversi modi, i seguenti comandi sono equivalenti tra loro:

```
assign ("nome.variabile", valore)
nome.variabile = valore
nome.variabile <- valore
valore -> nome.variabile
```

Dove `nome.variabile` può essere un qualunque nome a scelta e `valore` può essere un valore numerico, una testo racchiuso da doppie virgolette, il nome di un'altra variabile o il risultato di una operazione o di una qualunque funzione. Si noti la "direzionalità" dell'assegnazione, simbolicamente espressa dalla freccia realizzata con i simboli.

La variabile creata può essere di qualunque tipo (numerico, testuale, logico, ecc.) e il tipo di variabile dipende da ciò che gli è stato assegnato.

Un vettore è un insieme di dati omogenei, cioè dello stesso tipo. La creazione di un vettore si ottiene col comando:

```
c (10.1,15.2,6.4,3.2,18.7)
[1] 10.1 15.2  6.4  3.2 18.7
```

La funzione `c()` combina i valori presenti nel suo elenco in un vettore. Si noti che il separatore d'elenco è la virgola (','). Eseguendo quest'istruzione sarà semplicemente visualizzata una riga con il contenuto del vettore. Si noti che la riga incomincia con il numero uno tra parentesi quadre ('`[1]`'), questo indica che la riga mostra un insieme di dati a cominciare dal primo: se per mostrare tutti occorrono più righe, all'inizio di ciascuna viene indicato il numero dell'elemento con cui la riga comincia.
Il vettore naturalmente può essere memorizzato in un "oggetto", contraddistinto da un nome, in modo da poterlo riutilizzare. Ad esempio:

```
nome.vettore <- c (10.1,15.2,6.4,3.2,18.7) #assegnazione del vettore
nome.vettore #visualizzazione del contenuto del vettore
[1] 10.1 15.2  6.4  3.2 18.7
```

Un sottoinsieme d'elementi di un vettore può essere selezionato aggiungendo al nome del vettore un indice tra parentesi quadre. Ad esempio:

```
subset <- nome.vettore[2:4]
subset
[1] 15.2  6.4  3.2
```

restituisce un vettore contenente il secondo, il terzo e il quarto elemento del vettore di partenza.

Per ottenere un sottocampione casuale di dimensione nota (in questo caso 3 elementi) si usa la funzione:

```
subset <- sample(nome.vettore,3)
subset
[1] 10.1  3.2  6.4
```

Naturalmente il risultato ottenuto non è prevedibile. È possibile specificare come parametro `replace = TRUE` se rendere possibile o meno estrarre più volte lo stesso elemento. Questa funzione sarà di grande utilità per realizzare le permutazioni necessarie per alcuni test statistici.

1.5 Aritmetica coi vettori

Naturalmente sono immediatamente disponibili tutte le operazioni aritmetiche (`+,-,*,/,^`) e le principali funzioni logaritmiche e trigonometriche (`log, exp, sin, cos, tan, sqrt`). Le operazioni sono eseguite rispettando il classico ordine di priorità salvo che questo non sia modificato utilizzando le parentesi. Ad esempio:

```
raggio <- 21.35
p.greco <- 3.14159265358979
circonferenza <- raggio * 2 * p.greco
area <- p.greco * raggio^2
circonferenza
area
```

Eseguendo questa sequenza di istruzioni ci si rende anche conto che i risultati vengono visualizzati solo alla fine, quando si richiamano i rispettivi nomi di variabili.
I vettori possono essere utilizzati in qualunque espressione aritmetica. In questo caso l'operazione è applicata a tutti gli elementi del vettore. Essendo i vettori insiemi di valori, ad essi possono essere applicate anche funzioni statistiche (`min, max, length, mean, var, sum`). Esegui il seguente esempio:

```
campione <- c (72.6, 40.1, 59.6, 92.3, 65.4, 27.1, 86.4, 28.6, 5.4, 82.0)
campione^2
min(campione) #valore minimo
max(campione) #valore massimo
length(campione) #numero di valori
mean(campione) #media
median(campione) #mediana
range(campione) #restituisce un vettore di due elementi, il min e max
sd(campione) #deviazione standard
var(campione) #varianza
sum((campione - mean(campione))^2)/(length(campione)-1)
```

Si noti che per le funzioni statistiche, così come per tutte le funzioni, i dati e gli eventuali parametri sono racchiusi tra parentesi e che la funzione di per sé restituisce un valore, o un insieme di valori. Eseguendo queste istruzioni ci si rende conto che l'elevazione al quadrato è eseguita su tutti i valori del vettore, il risultato dell'operazione naturalmente è stato restituito, cioè visualizzato a schermo, ma non memorizzato in alcuna variabile (non viene assegnato). Infine gli ultimi due valori ottenuti sono identici: l'ultima operazione infatti è lo stesso calcolo della varianza eseguito dalla funzione `var()`.

La correlazione tra due vettori si ottiene con la funzione statistica:

```
cor(vettore1,vettore2)
```

Altre operazioni utili con i vettori sono:

```
sort(campione) #da gli stesse valori del vettore ma in ordine crescente.

order(campione) #restituisce l'ordine dei valori nel vettore.
```

Attenzione: quando si compiono operazioni tra due vettori, il primo elemento dell'uno viene operato con il primo elemento dell'altro, il secondo col secondo e così fino alla fine. Se un vettore è più corto dell'altro, quando termina viene riutilizzato dall'inizio. Si provi il seguente esempio:

```
a = c (10,10,10,10,10,10,10,10,10,10)
b = c (1,2)
a*b
[1] 10 20 10 20 10 20 10 20 10 20
```

Fra le innumerevoli possibilità di R vi è quella di creare facilmente delle sequenze regolari di numeri. Si veda i seguenti esempi:

```
c (1:30)
```

genera un vettore con la sequenza 1, 2, …, 30. La funzione:

```
rep (1,30)
```

genera un vettore in cui si ripete trenta volte il valore 1. Mentre la funzione:

```
seq (from=1, to=30, by=0.5)
```

genera il vettore da 1 a 30 con intervalli di 0.5. Si noti che la stessa funzione può essere scritta nel formato conciso:

```
seq (1,30,0.5)
```

La possibilità di utilizzare formati concisi è una caratteristica comune alla maggior parte delle funzioni che prevedono più parametri di funzionamento (in questo caso ci sono 5 possibili parametri, non tutti utilizzabili contemporaneamente; la cui spiegazione viene data digitando `?seq`). L'utilizzo dei formati concisi potrebbe generare però esecuzioni difformi da quelle desiderate ed è quindi da evitare se non si ha un'ottima padronanza della sintassi della funzione).

1.6 Operazioni logiche

I valori logici sono normalmente il risultato di operazioni logiche. Gli operatori logici utilizzabili in R sono:

`<`	minore
`<=`	minore o uguale
`>`	maggiore
`>=`	maggiore o uguale
`==`	uguale
`!=`	diverso
`&`	è l'intersezione ('e')
`|`	è l'unione ('o')
`!`	è la negazione ('non')

Per maggiore chiarezza si provi la seguente sequenza di istruzioni:

```
a <- c (1:30)
a <= 15
```

Le operazioni logiche eseguite su valori mancanti o comunque non utilizzabili, restituiscono il valore `NA`. Invece operazioni aritmetiche "impossibili", come ad esempio le divisioni per zero, restituiscono il valore `NaN` che significa che "non è un numero".

2 Panoramica sugli oggetti

Gli oggetti che possono essere maneggiati in R possono essere i seguenti:

variabili	contengono un singolo dato
vettori	insieme lineare di elementi omogenei per tipologia, tutti numeri, tutte stringhe, ecc.
fattori	vettore utilizzato per classificare o suddividere in "livelli" gli elementi di un altro vettore
array e matrici	sono vettori multimensionali, le matrici hanno due dimensioni: righe e colonne
liste	insieme di elementi disomogenei tra loro
data frame	sono matrici bidimensionali dove ogni colonna può avere una tipo di dato diverso dalle altre
funzioni	insieme di istruzioni che possono alla fine fornire un risultato di qualunque tipo, associato al nome stesso della funzione

Variabili e vettori sono già stati descritti, di seguito saranno brevemente illustrati gli altri oggetti.

2.1 Fattori

I "**fattori**" sono vettori utilizzati per classificare o suddividere in "livelli" gli elementi di un altro vettore di pari lunghezza. Un esempio classico è quello della suddivisione di una serie di misure eseguite in due siti, ad esempio "A" e "B". Per trasformare un vettore, contenente le etichette dei livelli, in "fattore" si usa la funzione `factor()`:

```
Fattore <- factor ( c ("A","A","A","A","A","B","B","B","B","B"))
Fattore
 [1] A A A A A B B B B B
Levels: A B
```

Per ottenere l'elenco dei livelli presenti nel fattore è possibile utilizzare la funzione:

```
levels (Fattore)
[1] "A" "B"
```

Il "fattore" consente di applicare una funzione a un vettore di dati suddivisi per livelli, allo scopo si usa la funzione `tapply()`, il seguente esempio aiuterà a comprendere: supponiamo di contare il numero di specie presenti in cinque campioni presi dal sito A e in cinque campioni del sito B e di voler ottenere il valore medio di specie per sito

```
fattore <- factor ( c ("A","A","A","A","A","B","B","B","B","B"))
dati <- c (2,3,7,8,9,35,37,31,38,36)
tapply (dati,fattore,mean)
   A    B
 5.8 35.4
tapply (dati,fattore,sd)
       A        B
3.114482 2.701851
```

2.2 Array e Matrici

Gli array sono vettori multidimensionali. L'esempio più semplice è quello delle matrici, cioè array bidimensionali rappresentabili come una tabella con un certo numero di righe e colonne. Per poter utilizzare una matrice è necessario specificarne le dimensioni col la funzione `dim()`, ad esempio:

```
dati <- c(1,4,6,32,6,7,4,6,8,5,3,6,7,67,4)
righe <- 5
colonne<- 3
matrice <- array (data=dati,dim=c(righe,colonne))
matrice
     [,1] [,2] [,3]
```

```
[1,]     1     7     3
[2,]     4     4     6
[3,]     6     6     7
[4,]    32     8    67
[5,]     6     5     4
```

Nel caso di un array le dimensioni possono essere più di due, nel caso specifico di una matrice, quindi con due sole dimensioni (righe e colonne), si può direttamente usare l'istruzione:

```
matrice <- matrix (data=dati,nrow=righe,ncol=colonne)
```

Un sottoinsieme d'elementi di un array o di una matrice può essere selezionato aggiungendo al nome della matrice gli indici tra parentesi quadre. Ad esempio:

```
subset <- matrice[4:5,2]
subset
[1] 8 5
```

2.3 Liste

Sono un insieme di elementi disomogenei tra loro. Si crea specificando un nome per ciascun elemento. Ogni elemento a sua volta può essere un qualunque tipo di oggetto, ad esempio

```
lista<-list(marito="Fred", moglie="Mary", figli=3, anni.figli=c(2,3,6))
lista
$marito
[1] "Fred"

$moglie
[1] "Mary"

$figli
[1] 3

$anni.figli
[1] 2 3 6
```

Nel caso sia omesso il nome, l'elemento è identificato da un numero progressivo. Per richiamare un elemento di una lista si utilizza il nome della lista e dell'elemento uniti dal simbolo `$`, oppure il numero progressivo corrispondente tra parentesi quadre, ad esempio:

```
lista$moglie
[1] "Mary"
lista[2]
[1] "Mary"
```

2.4 Data frame

Nella loro forma più semplice posso essere considerati come matrici dove ogni colonna può avere una tipo di dato diverso dalle altre. È il modo più pratico per maneggiare tabelle di dati. Come per le liste è possibile specificare un nome per ciascuna colonna, nel caso in cui venga omesso il nome viene utilizzato quello dell'oggetto che costituisce la colonna. Ad esempio:

```
fattore <- factor ( c ("A","A","A","A","A","B","B","B","B","B"))
specie <- c (2,3,7,8,9,35,37,31,38,36)
individui <- c (321,654,765,865,964,353,372,315,385,364)
dataset <- data.frame (siti=fattore, specie, individui)
dataset
```

```
   siti specie individui
1     A      2        321
2     A      3        654
3     A      7        765
4     A      8        865
5     A      9        964
6     B     35        353
7     B     37        372
8     B     31        315
9     B     38        385
10    B     36        364
```

Si noti che solo per la prima colonna è stato assegnato un nome. Si noti anche che come prima colonna è stato utilizzato un oggetto fattore. Anche per i data frame è possibile richiamare un elemento, in questo caso una colonna di dati, utilizzando la notazione `$`, cioè:

```
dataset$siti
 [1] A A A A A B B B B B
Levels: A B
```

È però possibile rendere più agevole l'uso delle colonne facendo sì che R riconosca la loro intestazione. Il comando è:

```
attach(dataset)
```

ora è possibile richiamare ciascuna colonna direttamente col suo nome:

```
siti
 [1] A A A A A B B B B B
Levels: A B
```

2.5 Trasposizione

È possibile trasporre una matrice o un data frame, cioè scambiare le colonne con le righe, con la funzione:

```
trasposto <- t(originale)
```

Se la matrice originale ha dimensioni AxB si otterrà una matrice di dimensioni BxA. Se si applica la funzione ad un vettore si ottiene una matrice con una sola colonna.

2.6 Importare un file di dati

In genere i dati che si vogliono analizzare sono memorizzati in tabelle di database o di fogli elettronici. R può importare i dati soltanto da file di testo "delimitato da tabulazioni", dove i dati appartenenti a colonne successive sono separati da un carattere di tabulazione e le righe sono separate da un "a capo" (fine paragrafo). Tutti i database e fogli di calcolo consentono di esportare una tabella di dati nel formato descritto. Una volta generato il file di testo, la tabella può essere importata col comando:

```
tabella <- read.table(file.choose())
```

In questo modo appare una finestra di dialogo che consente di identificare il file contenente i dati. Attenzione però, anche l'ultima riga deve essere terminata, non tutti i programmi generano correttamente il file di testo e in questo caso si genera il messaggio di errore:

```
Warning message:
incomplete final line found by readTableHeader on …
```

Per risolvere il problema è necessario aggiungere un a capo alla fine de file e per farlo è possibile utilizzare un qualunque programma editor di testo.

Se la prima riga della tabella contiene l'intestazione delle colonne la sequenza comando diventa:

```
tabella <- read.table(file.choose(),header=T)
attach(tabella)
```

Infine se alcune colonne rappresentano dei fattori d'analisi è necessario per ciascuno creare l'oggetto "fattore" corrispondente. Un esempio aiuterà a comprendere. Consideriamo di aver analizzato l'abbondanza di 4 specie in cinque campioni prelevati in due siti e in due date. La tabella dei dati è:

Sito	Data	Replica	Specie1	Specie2	Specie3	Specie4
A	1	1	36	6	58	78
A	1	2	33	92	11	1
A	1	3	54	1	69	51
A	1	4	81	29	55	87
A	1	5	31	6	80	77
A	2	1	67	86	90	55
A	2	2	87	64	20	36
A	2	3	51	83	29	45
A	2	4	99	66	12	8
A	2	5	30	29	73	72
B	1	1	5	55	52	19
B	1	2	39	45	23	88
B	1	3	91	74	29	49
B	1	4	30	97	29	80
B	1	5	52	8	84	81
B	2	1	64	98	29	81
B	2	2	11	24	97	70
B	2	3	70	92	36	53
B	2	4	33	65	92	43
B	2	5	42	78	97	23

Dopo aver salvato la tabella in un file di testo delimitato da tabulazioni, si procede in R come segue:

```
tabella.df <- read.table(file.choose(),header=T) #scegliere il file
attach(tabella.df)
Data<-factor(Data) #questo serve solo per fattori espressi da numeri
tabella.df
   Sito Data Replica Specie1 Specie2 Specie3 Specie4
1     A    1       1      36       6      58      78
2     A    1       2      33      92      11       1
3     A    1       3      54       1      69      51
4     A    1       4      81      29      55      87
5     A    1       5      31       6      80      77
6     A    2       1      67      86      90      55
7     A    2       2      87      64      20      36
8     A    2       3      51      83      29      45
9     A    2       4      99      66      12       8
10    A    2       5      30      29      73      72
11    B    1       1       5      55      52      19
12    B    1       2      39      45      23      88
13    B    1       3      91      74      29      49
14    B    1       4      30      97      29      80
15    B    1       5      52       8      84      81
16    B    2       1      64      98      29      81
17    B    2       2      11      24      97      70
18    B    2       3      70      92      36      53
19    B    2       4      33      65      92      43
20    B    2       5      42      78      97      23
```

Tutte le colonne che contengono testo sono automaticamente considerate come fattori. A questo punto si può lavorare sui dati importati. Ad esempio volendo la media dell'abbondanza della specie 1 nei due diversi siti si può usare l'istruzione:

```
tapply (Specie1,Sito,mean)
   A    B
56.9 43.7
```

3 Principi di programmazione

Con R è possibile anche scrivere veri e propri programmi, memorizzarli e riutilizzarli all'occorrenza. Anche se l'approfondimento di quest'argomento esula dagli scopi di queste note, una conoscenza dei principi di base sarà utile per la realizzazione di semplici funzioni e per poter utilizzare i numerosi programmi già disponibili.

In generale è possibile raggruppare insieme più comandi racchiudendoli tra parentesi graffe. Inoltre come in tutti i linguaggi di programmazione è disponibile la funzione condizionale:

```
if (expr1) expr2 else expr3
```

dove la prima espressione deve restituire un valore logico `TRUE` o `FALSE` e determina l'esecuzione o meno delle espressioni successive.

Sono naturalmente disponibili sia cicli incondizionati sia condizionati:

```
for (name in expr1) expr2

repeat expr

while (condition) expr
```

Per maggiori in formazioni si rimanda alla documentazione fornita con R.

3.1 Creare nuove Funzioni

È possibile in qualunque momento definire una nuova funzione e darle un nome. Naturalmente ogni funzione deve essere definita prima di poterla utilizzare. Nella definizione di funzione occorre specificare tra parentesi l'elenco dei dati e dei parametri che dovranno essere forniti alla funzione. Infine il nome stesso della funzione viene utilizzato per restituire il "risultato" della funzione. Ecco un semplice esempio per creare la funzione errore standard dei dati contenuti in un vettore:

```
es <- function (vettore) {
deviazione.standard<-sd(vettore)
n<-length(vettore)
es<-deviazione.standard/sqrt(n) #assegna il risultato al nome alla funzione
}
```

Per verificarne il funzionamento eseguire:

```
campione <- c (72.6, 40.1, 59.6, 92.3, 65.4, 27.1, 86.4, 28.6, 5.4, 82.0)
errore.standard<-es(campione)
errore.standard
[1] 9.245927
```

Si noti che gli oggetti definiti e maneggiati all'interno della funzione appartengono ed essa, in pratica "esistono" solo durante l'esecuzione della funzione stessa. Il loro contenuto scomparirà al termine dell'esecuzione. Eventuali altri oggetti presenti in memoria con lo stesso nome non saranno toccati. L'unico oggetto restituito sarà esclusivamente quello associato al nome della funzione.

3.2 Utilizzo di funzioni e librerie esterne

In R sono disponibili numerose librerie di funzioni e programmi utilizzabili dagli utenti. Inoltre gli sviluppatori sono continuamente all'opera per creare nuove librerie. Le librerie sono archiviate in un'apposita sottocartella del programma R, chiamata "library". La libreria principale, sempre disponibile si chiama "base". Fra le altre librerie vi è ad esempio la libreria "mva" contenente strumenti classici per l'analisi multivariata come PCA, similarità, ecc.

Per utilizzare una libreria occorre prima specificarne l'uso con la funzione:

```
library(nomelibreria)
```

Il modo più semplice per memorizzare le funzioni per poterle riutilizzare in seguito, è di salvarle in un file di testo (generalmente denotato con l'estensione .R). Queste funzioni possono essere caricate in memoria con il comando:

```
source("nome.del.file") #impostare prima la directory nel menu "file"
```

oppure più semplicemente con

```
source(file.choose()) #scegliere il file desiderato
```

Una volta caricate le funzioni, queste possono essere utilizzate direttamente tutte le volte che si vuole.

4 Grafici in R

R consente di realizzare, con semplicità, grafici di qualità professionale. Questi sono poi esportabili come file in numerosi formati (sia raster sia vettoriali) o direttamente copiabili per l'utilizzo in altri programmi.
Per la realizzazione dei grafici si utilizzano delle funzioni specifiche nelle quali occorre specificare sia i dati da utilizzare sia eventuali parametri opzionali.

4.1 Grafici XY ed istogrammi a barre

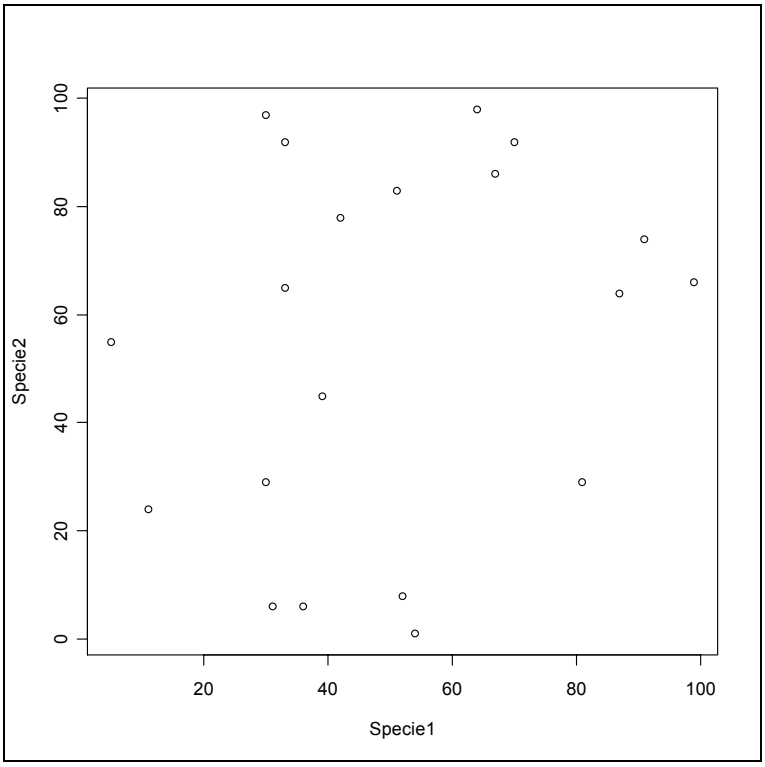
Per disegnare un grafico dei punti, note le coordinate X Y, si usa la funzione grafica:

```
plot(X,Y)
```

dove `X` e `Y` sono due vettori di uguale lunghezza.

Utilizzando i dati dell'esempio di importazione di un file esterno, si ottiene

```
tabella.df <- read.table(file.choose(),header=T) #scegliere il file
attach(tabella.df)
Data<-factor(Data)
plot(Specie1,Specie2)
```



La sintassi della funzione è:

```
plot(X,Y,xlim=…,ylim=…, type=…, main=…, xlab=…, ylab=…, ...)
```

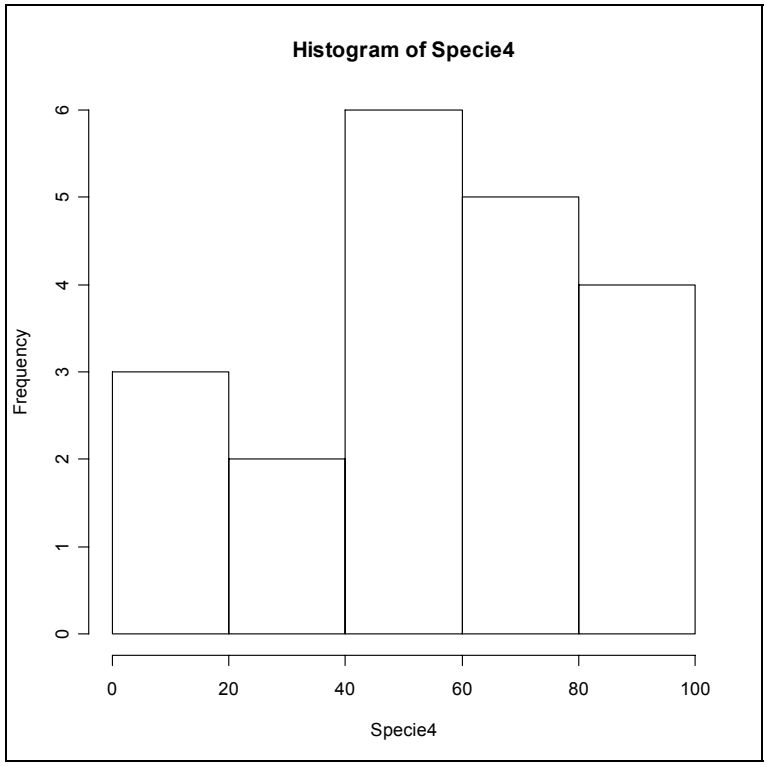
dove `xlim` e `ylim` sono i range degli assi (occorre fornire un vettore col limite min e max), `type` identifica il tipo di simbolo ("p" punti, "l" linee, "b" punti e linee, "h" linee verticali o istogramma, ecc.) , `main` è una stringa col titolo principale, `xlab` e `ylab` sono stringhe con le etichette da dare agli assi.

4.2 Istogrammi di frequenza

Per realizzare un istogramma di frequenza di un dato vettore si usa la funzione grafica:

```
hist(vettore)
```

Utilizzando i dati della specie 4 del precedente esempio si ottiene:



La sintassi della funzione è:

```
hist(vettore, xlab=…, ylab=…, main=…, breaks=…)
```

dove `main` è una stringa col titolo principale, `xlab` e `ylab` sono stringhe con le etichette da dare agli assi, `breaks` è il numero di classi.

4.3 Grafici Box-Whisker

Per realizzare questi grafici si usa la funzione grafica:

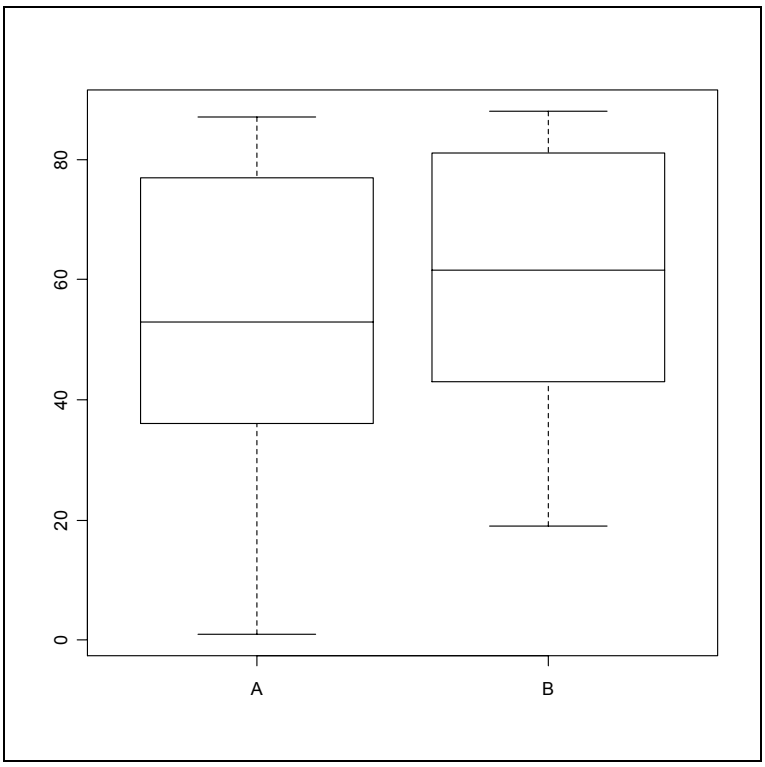
```
boxplot (vettore)
```

Questo grafico descrive la tendenza centrale della distribuzione mediante il valore mediano, la dispersione dei dati è rappresentata dai quartili (25° e 75° percentile) e l'estensione complessiva del campione.

Nel caso il vettore contenga dati relativi a diversi livelli, così come specificato da un dato fattore, è possibile ottenere un grafico per ciascun livello utilizzando la notazione:

```
boxplot (vettore~fattore)
```

Ad esempio, utilizzando i dati della Specie 4 del precedente esempio e il fattore Sito si ottiene:



I parametri di rappresentazione possono essere modificati. Per maggiori informazioni si consulti la documentazione fornita con R.

4.4 Comandi grafici

È possibile utilizzare alcuni comandi per aggiungere informazioni o modificare un grafico esistente. I più comuni sono:

```
points(x, y) # aggiunge un punto nelle coordinate x y
lines(x, y) # connette linee
text(x, y, labels, ...) # aggiunge una etichetta al punto di coordinate x y
abline(a, b) # aggiunge una retta di pendenza b e intercetta a
polygon(x, y, ...) # disegna un poligono
legend(x, y, legend, ...) # aggiunge una legenda
title(main, sub) # aggiunge un titolo
axis(side, ...) # aggiunge un asse
```

Per i dettagli si rimanda alla documentazione fornita con R ed in particolare alla guida introduttiva.

5 Statistica in R

Quando si presuppone che le analisi statistiche siano condotte su dati memorizzati in file esterni delimitati da tabulazioni e dotati d'intestazione, la procedura d'importazione viene riportata all'inizio dell'esempio.

5.1 Distribuzioni statistiche

R è in grado di generare set di numeri casuali estratti ad esempio da una distribuzione normale di cui sono dati alcuni parametri. Il comando più semplice è:

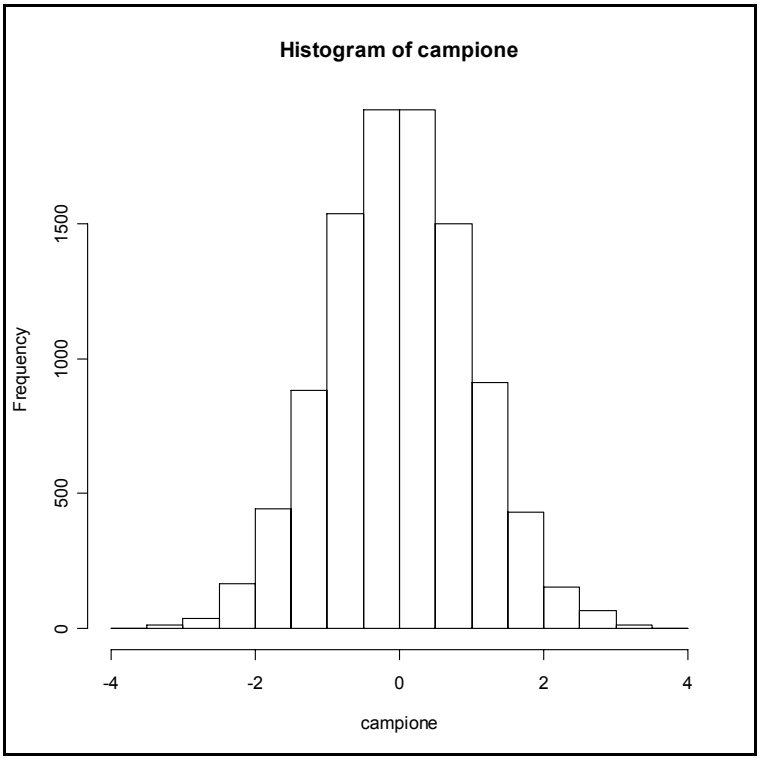
```
rnorm(n, mean=0, sd=1)
```

dove n è il numero di valori che verrà restituito a partire da una popolazione con media 0 e deviazione standard 1. Si provi il seguente esempio:

```
campione = rnorm(50, mean=0, sd=1)
campione
mean(campione)
sd(campione)
```

Le stime di media e deviazione standard, ottenute dal campione, non saranno esattamente uguali a quelli della popolazione di partenza, ma se si ripete la prova incrementando la dimensione del campione si osserveranno valori sempre più vicini a quelli "reali".
Per verificare graficamente la distribuzione di frequenza è possibile utilizzare il comando:

```
hist(campione)
```



Le distribuzioni statistiche disponili in R sono:

Distribuzione	**nome in R**	**argomenti addizionali**
beta	beta	shape1, shape2, ncp
binomiale	binom	size, prob
Cauchy	cauchy	location, scale
chi-quadro	chisq	df, ncp
esponenziale	exp	rate
F	f	df1, df1, ncp
gamma	gamma	shape, scale
geometrica	geom	prob
hypergeometrica	hyper	m, n, k
log-normale	lnorm	meanlog, sdlog
logistica	logis	location, scale
binomiale negativa	nbinom	size, prob
normale	norm	mean, sd
Poisson	pois	lambda
t di Student	t	df, ncp
uniforme	unif	min, max
Weibull	weibull	shape, scale
Wilcoxon	wilcox	m, n

Aggiungendo al nome della distribuzione i seguenti prefissi, si ottiene:

d	densità
p	funzione di distribuzione
q	quantile
r	numeri casuali estratti dalla distribuzione

Per maggiori in formazioni si rimanda alla documentazione fornita con R.

5.2 Intervallo di confidenza e confronto delle medie

Per calcolare l'intervallo di confidenza della media, dato un vettore e un determinato livello di confidenza, si può utilizzare la funzione:

```
t.test(vettore, var.equal=T,conf.level=.95)
```

Utilizzando i dati della Specie1 forniti nell'esempio di importazione dati, si ottiene

```
t.test(Specie1,var.equal=T,conf.level=.95)

        One Sample t-test

data:  Specie1
t = 8.6057, df = 19, p-value = 5.575e-08
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 38.06634 62.53366
sample estimates:
mean of x
     50.3
```

La stessa funzione può essere utilizzata per eseguire il t-test per il confronto delle medie di due campioni. Ad esempio generiamo due campioni casuali a partire da due popolazioni con distribuzione normale, uguale varianza ma differenti medie, e confrontiamoli tra loro:

```
campione1 = rnorm(50, mean=0, sd=1)
campione2 = rnorm(50, mean=10, sd=1)
t.test(campione1, campione2,var.equal=T,conf.level=.95)

        Two Sample t-test

data:  campione1 and campione2
t = -51.5435, df = 98, p-value = < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -10.425492  -9.652475
sample estimates:
 mean of x  mean of y
-0.1262038  9.9127795
```

Il valore di probabilità (`p-value`) inferiore al limite prefissato di 0.05 ci porta a rigettare l'ipotesi di nulla di nessuna differenza tra le medie. La stessa funzione può essere utilizzata anche nel caso di popolazioni con varianze differenti (ponendo `var.equal=F`), in questo caso viene applicato la versione di Welch del t-test.
Si noti che la funzione t.test fornisce un risultato composto da numerose informazioni. L'intero risultato può essere assegnato ad un oggetto. Di fatto si tratta di una "lista" dove ogni informazione ha un suo nome e quindi può essere richiamata. Ad esempio se si desidera avere esclusivamente il valore di probabilità:

```
risultato <- t.test(campione1, campione2,var.equal=T,conf.level=.95)
risultato$p.value
[1] 8.299888e-73
```

Per un elenco completo dei valori disponibili si consulti l'help della funzione t.test sotto la voce 'value'.

5.3 Modello lineare

In R è possibile creare dei modelli statistici. Il caso più semplice e maggiormente utilizzato è quello dei modelli lineari, il più semplice dei quali si identifica con l'equazione di una retta:

$$y = b + a\,x$$

Il modello lineare nella sua notazione più semplice è:

```
lm(formula)
```

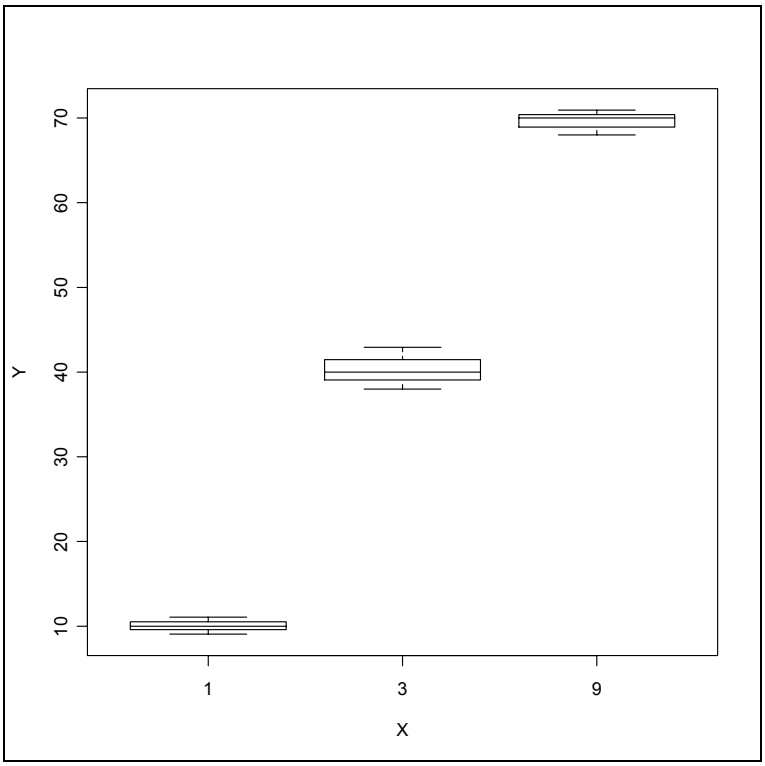
dove *formula* è una descrizione simbolica del modello. Questa si compone di due parti:

```
response ~ terms
```

Il primo indica la variabile dipendente (y) mentre il secondo comprende tutti i termini predittori del modello. Di fatto il primo termine è dato da un vettore che contiene tutti i valori misurati, mentre la seconda parte è costituita dai fattori che entrano in gioco. Un esempio chiarirà l'uso:

```
X=factor(c(a,a,a,b,b,b,c,c,c)) #fattore a 3 livelli, 3 repliche ciascuno
Y=c(10,11,9,40,43,38,70,68,71) #risultati conseguiti
lm(Y~X)
Call:
lm(formula = Y ~ X)

Coefficients:
(Intercept)           X3           X9
      10.00        30.33        59.67
plot(Y~X)
```



Modelli lineari complessi a più fattori si costruiscono con la seguente notazione:

`fattore1 + fattore2`	fattori ortogonali
`fattore1:fattore2`	interazione di due fattori ortogonali
`fattore1%in%fattore2`	il secondo fattore è *nested* nel primo

Per l'esempio dell'abbondanza di una specie campionata in due siti e in due date il modello lineare è:

```
lm(Specie1 ~ Sito + Data + Sito:Data)
```

Questa è espressione può essere condensata con la notazione compatta:

```
lm(Specie1 ~ Sito*Data)
```

5.4 Analisi della Varianza

Dato un modello lineare, l'analisi della varianza può essere condotta semplicemente con la funzione:

```
anova(linear.model)
```

Utilizzando l'esempio dell'abbondanza della Specie2 campionata in due siti e in due date:

```
tabella.df <- read.table(file.choose(),header=T) #scegliere il file
attach(tabella.df)
Data<-factor(Data)

linear.model <- lm(Specie2 ~ Sito + Data + Sito:Data)
results <- anova(linear.model)
results
```

```
Analysis of Variance Table

Response: Specie2
          Df  Sum Sq Mean Sq F value  Pr(>F)
Sito       1  1513.8  1513.8  1.5386 0.23271
Data       1  3699.2  3699.2  3.7598 0.07034 .
Sito:Data  1   672.8   672.8  0.6838 0.42043
Residuals 16 15742.0   983.9
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

La tabella ANOVA visualizzata sulla "console" può essere esportata ad esempio selezionandola col mouse, copiandola e incollandola in un file di testo (utilizzando ad esempio il notepad di windows) o direttamente nel documento desiderato. Il file di testo può poi essere eventualmente importato in un foglio elettronico (ad esempio Microsoft Excel) come testo a larghezza fissa o delimitato da spazi, in questo modo possono essere riutilizzati tutti i valori della tabella.

Questo esempio può essere esteso a qualunque modello lineare. Attenzione però! In R i valori di F attesi sono sempre calcolati dividendo il "`Mean Sq`" di ciascun fattore per il "`Mean Sq`" del residuo. Per eseguire il test corretto con modelli complessi è quindi necessario determinare i corretti denominatori per ciascun fattore, calcolare il valore atteso di F e ottenere il valore di probabilità associato in funzione dei rispettivi gradi di libertà.

Nel caso specifico, considerando che i siti e le date siano stati scelti casualmente, il denominatore corretto per i l'analisi dei fattori principali è dato dall'interazione. Dell'analisi eseguita da R quindi possiamo utilizzare i valori MS coi rispettivi gradi di libertà e sulla base di questi calcolare i corretti valori di F attesi e le rispettive probabilità. Si può operare così:

```
MS <- results$"Mean Sq" # estrazione dei dati
DF <- results$"Df"
F.Sito <- MS[1]/MS[3] # calcolo di F
F.Data <- MS[2]/MS[3]
p.val.Sito <- 1-pf(F.Sito,DF[1],DF[3]) # funzione di probabilità F
p.val.Data <- 1-pf(F.Data,DF[2],DF[3])
p.val.Sito
        1
0.3743341
p.val.Data
        2
0.2566314
```

La determinazione dei test corretti per disegni complessi di ANOVA esula dagli scopi di questa nota e si rimanda ai manuali di statistica.

Indice analitico

T

U

V

W

Bibliografia

Ihaka, R. and Gentleman, R. (1996). "R: A language for data analysis and graphics." Journal of Computational and Graphical Statistics 5, 299-314.

Venables, W. N. and Smith, D. M. (2002). "An Introduction to R." R Development Core Team, Auckland. 105 pp.