Introduzione all'analisi statistica con **R**

Riccardo Massari

riccardo.massari@uniroma1.it

1 Introduzione

1.1 Generalità

1.1.1 Che cos'è R

R è un "ambiente" basato sul linguaggio S, per la gestione e l'analisi statistica di dati e la produzione di grafici, disponibile gratuitamente sotto i vincoli della GPL (General Public Licence). Informazioni, risorse e molto altro sono disponibili al sito:
`http://cran.r-project.org/`

La versione più recente di **R** può essere scaricata dalla pagina:
`http://cran.r-project.org/bin/windows/base/`

Una volta scaricato il *file* `.exe` sul proprio computer è sufficiente lanciare il programma di installazione. Sono comunque presenti dettagliate spiegazioni sul sito di distribuzione del programma.

In **R** l'analisi è fatta attraverso una serie di passi, con risultati intermedi che sono immagazzinati in "oggetti". Rispetto a programmi come SAS o SPSS, **R** fornisce un *output* minimo. I rimanenti risultati possono essere successivamente richiamati tramite opportuni comandi.

Per approfondimenti su aspetti che non sono trattati in queste dispense si rimanda a:

1. Iacus, Masarotto *Laboratorio di Statistica con* **R**, McGraw-Hill Italia, 2003.
2. Crivellari *Analisi statistica dei dati con* **R**, Apogeo, 2006.

oltre alle numerose dispense per approfondire diversi aspetti di **R**, disponibili in rete. In particolare, nel sito `http://cran.r-project.org/other-docs.html`, si possono trovare anche numerose dispense in italiano (sezione "Other languages").

1.1.2 Primi passi

Avviato il programma **R**, apparirà una finestra (*Console*) con il simbolo di *prompt*
`>`

che indica che l'ambiente è pronto per ricevere delle istruzioni, che vengono immesse scrivendo il comando sulla riga che comincia dal simbolo di *prompt* (detta linea di comando).

Una volta concluso il comando si digita il tasto di "Invio". Se il comando è completo, comparirà una nuova linea di comando con il simbolo di *prompt*. Altrimenti comparirà una nuova riga con il simbolo

```
+
```

che sta a significare che il comando deve essere completato.

La console di **R** può essere utilizzata come semplice calcolatrice. E' un ambiente interattivo, ossia i comandi producono una risposta immediata. Ad esempio, se scriviamo `2 + 2`, comparirà nella riga successiva il risultato. Alcuni esempi:

```
> #somma
> 2 + 2

[1] 4

> #moltiplicazione
> 2 * 2

[1] 4

> #sottrazione e divisione
> (2 - 3)/6

[1] -0.167

> #ATTENZIONE: è diverso da scrivere
> 2 - 3/6

[1] 1.5

> #elevamento a potenza
> 2^2

[1] 4
```

Osservazione Tutto quello che viene scritto su una riga di comando dopo `#` non viene letto dal programma. In tale modo, è possibile inserire dei commenti alle operazioni che vengono effettuate.

Esistono poi diverse "funzioni" (vedi più avanti) per le principali operazioni matematiche e trigonometriche. Per dare solo qualche esempio, `sqrt` serve per estrarre la radice quadrata da un numero, `log` calcola il logaritmo naturale ed `exp` l'esponenziale:

```
> sqrt(9)
```

```
[1] 3

> log(1)

[1] 0

> exp(0)

[1] 1
```

Gli operatori `==` (uguale), `>=/<=` (maggiore/minore uguale), `>/<` (maggiore/minore), `!=` (diverso) sono utilizzati per effettuare confronti tra oggetti. Sono impiegati in diverse situazioni, ad esempio, come si vedrà più avanti, nella selezione di elementi da un vettore o da una matrice dei dati. Il confronto tra due elementi produce il risultato `TRUE` se il confronto è verificato, `FALSE` altrimenti. Alcuni esempi:

```
> 2 * 2 == 4

[1] TRUE

> 2 * 2 > 4

[1] FALSE

> 2 * 2 >= 4

[1] TRUE

> 2 * 2 != 4

[1] FALSE
```

Ogni entità che il programma crea e manipola è definita un **oggetto**, che può essere un numero, una variabile, una funzione, o più in generale, strutture costruite a partire da tali componenti.

Gli oggetti creati dall'utilizzatore del programma durante una sessione di lavoro possono essere salvati con un nome, per poter essere riutilizzati durante la sessione. Per assegnare un nome ad un oggetto si impiega il comando `<-`, o, più semplicemente `=` .

```
> x <- 4
> x

[1] 4

> y = 3 * 2
> y
```

```
[1] 6

> #è possibile effettuare delle operazioni con gli oggetti salvati
> x * y

[1] 24

> #e creare nuovi oggetti da salvare con un nome
> z = (y - x) * 2
> z

[1] 4
```

I nomi degli oggetti che vengono creati e salvati dall'utente possono contenere un qualunque carattere alfa-numerico. Non devono mai iniziare con un numero, non ci devono essere spazi vuoti, né gli operatori per le operazioni matematiche. Sarebbe opportuno, inoltre, che non coincidano con nomi di funzioni utilizzate dal programma.

```
> #i nomi degli oggetti non devono contenere spazi vuoti
> pro va = 2
Error: unexpected symbol in "pro va"

> #non devono contenere simboli matematici
> #creano confusione
> x-y = 2
Error in x - y = 2 : could not find function "-<-"

> #non devono iniziare con un numero
> 2x = 2
Error: unexpected symbol in "2x"

> #alcuni nomi sono "protetti" da R
> for = 3
Error: unexpected '=' in "for ="
> break = 5
Error in break = 5 : invalid (NULL) left side of assignment
```

Si osservi che, anche se i nomi degli oggetti non devono iniziare con un numero, possono essere composti anche da numeri:

```
> x2 = 5
> x2

[1] 5
```

Occorre fare attenzione al fatto che **R** è *case sensitive*, nel senso che distingue tra maiuscole e minuscole. Ad esempio, `Y` e `y` si riferiscono ad oggetti diversi.

```
> y
[1] 6
> Y
Error: object 'Y' not found
```

Infine, se si attribuisce a due oggetti diversi lo stesso nome, **R** cancellerà il primo dei due oggetti e manterrà in memoria solo il secondo:

```
> a = 2
> a

[1] 2

> a = 4
> a

[1] 4
```

1.2 Gli oggetti in R

1.2.1 Spazio di lavoro

Gli oggetti creati dall'utente vengono temporaneamente salvati nello **spazio di lavoro** (*workspace*). Per sapere qual è lo spazio di lavoro di *default*, ossia quello utilizzato dal programma nel momento in cui viene lanciato, si utilizza il comando `getwd()`, che mostrerà il *path* della cartella nel computer che è stata automaticamente selezionata al momento dell'installazione del programma. Si può lavorare nel *workspace* di *default*, ma è consigliabile utilizzare spazi di lavoro diversi, e quindi cartelle diverse, per lavori diversi.

Per modificare lo spazio di lavoro, si utilizza il comando `setwd()`. Ad esempio, se è stata creata una directory dal nome "lezioniR" nella directory "C:", per salvare tutto il lavoro in questa cartella, all'inizio della sessione di lavoro si scriverà[1]:

```
setwd("C:/lezioniR")
```

In alternativa, dal menu "File", si sceglie l'opzione "Cambia directory" e si sceglie la la cartella.

In questa cartella devono essere inseriti tutti i dati, provenienti da *file* esterni ad **R** che si vogliono impiegare nella sessione di lavoro. Inoltre, in questa cartella verranno salvate le figure, le matrici di dati e, più in generale, tutti gli oggetti che vengono creati durante la sessione di lavoro.

A quest'ultimo riguardo, come già accennato, gli oggetti vengono salvati solo temporaneamente nello spazio di lavoro, fino alla fine della sessione, e non sono visibili "fisicamente" nella cartella selezionata come *workspace*. Per salvare tutto lo spazio di lavoro in maniera definitiva, si utilizza il comando `save.image()`, che crea un *file* senza nome nella cartella

[1] Fate attenzione all'orientamento dello *slash*, che è opposto da quello impiegato, ad esempio, sotto Windows.

selezionata con estensione `.Rdata`, la cui icona è identica a quella di **R**. Se invece si vuole salvare lo spazio di lavoro con un nome si scriverà `save.image("nomefile.Rdata")`.

Infine, si possono salvare solo alcuni oggetti presenti nello spazio di lavoro, con il comando `save()`. Alcuni esempi:

```
save(x, file = "x.Rdata")
save(x, y, z, file = "prova.Rdata")
```

Nel primo caso si salva solo l'oggetto `x`, che si troverà nel *file* `x.Rdata` visibile nella cartella indicata come spazio di lavoro, nel secondo gli oggetti `x, y, z`.

Ciascuno di questi *file* può essere importato in una nuova sessione sessione semplicemente avviando **R** dal *file* `.Rdata` di interesse, oppure utilizzando il comando load():

```
load(".Rdata")
load("prova.Rdata")
```

Da quanto scritto sopra, dovrebbe essere chiaro che nel primo caso si carica tutto lo spazio di lavoro precedentemente salvato[2], nel secondo solo gli oggetti `x, y, z`.

Il comando `ls()` permette di visualizzare i nomi degli oggetti memorizzati. Per eliminare uno o più oggetti dallo spazio di lavoro si utilizza il comando `rm()`, mentre il comando `rm(list = ls())` elimina tutti gli oggetti nello spazio di lavoro. Esempio:

```
> ls()

[1] "a"  "x"  "x2" "y"  "z"

> #cancella solo uno o più oggetti
> rm(x)
> ls()

[1] "a"  "x2" "y"  "z"

> rm(y, x2)
> ls()

[1] "a" "z"

> #cancella tutti gli oggetti nello spazio di lavoro
> rm(list = ls())
> ls()

character(0)
```

[2]Anche le operazioni di salvataggio e di importazione di un *workspace* può essere effettuata con il menu "File".

1.2.2 Funzioni

Tra gli oggetti in **R**, un ruolo di particolare rilevanza è rivestito dalle **funzioni**, insiemi di comandi elementari, che permettono di effettuare dalle più comuni operazioni matematiche, alle più sofisticate tecniche statistiche, e altro ancora. I comandi incontrati fino ad ora, come `save.image`, `ls`, o `sqrt`, sono esempi di funzioni.

La sintassi di una generica funzione è del tipo:
`nomefunzione(argomento1, argomento2, argomento3,...)`
dove gli argomenti di una funzione, separati da una virgola, possono essere oggetti come vettori, matrici, altre funzioni, parametri o operatori logici. Alcuni argomenti possono avere dei valori di *default*, se sono posti uguali ad un valore, o ad una condizione, ecc.

Non è sempre necessario specificare tutti gli argomenti di una funzione. Anche l'ordine degli argomenti non è importante. E' necessario ricordare che, qualora gli argomenti della funzione siano inseriti in un ordine diverso da quello definito, si deve sempre richiamare il nome dell'argomento che si vuole impiegare. Per richiamare l'*help* in linea su una funzione, e quindi avere informazioni sulla sintassi della funzione, oltre a esempi pratici di impiego, si scrive:
`?nomefunzione`.

Osservazione Quali argomenti vadano sempre specificati e quali solo in base alle finalità dell'analisi è un argomento complesso da trattare in astratto, senza esempi pratici. Nel seguito verranno introdotte numerose funzioni, con esempi che permetteranno di chiarire, di volta in volta, quali argomenti vadano specificati e quali no.

1.2.3 Pacchetti

Le funzioni di **R** sono organizzate in **pacchetti**, i più importanti dei quali sono già disponibili quando si accede al programma. Alcuni pacchetti, pur essendo presenti nella *release* di base di **R**, devono essere caricati durante la sessione di lavoro, mediante il comando:
`require(nomepacchetto)`

Per sapere quali sono i pacchetti già presenti nella *release* di **R** con cui si sta lavorando, basta scrivere:
`library()`

In alternativa, dal menu "Pacchetti" si sceglie l'opzione "Carica pacchetto". In tal caso apparirà una finestra con la lista dei pacchetti già installati. Sarà quindi sufficiente selezionare il nome del pacchetto da caricare con il *mouse* e poi cliccare su "OK".

Per avere informazioni sul pacchetto e sulle funzioni in esso contenute:
`help(package = "nomepacchetto")`

Alcuni pacchetti non sono invece presenti nella *release* di base di **R**. Per installare un pacchetto non presente, è sufficiente scrivere:
`install.packages("nomepacchetto")`

Table 1: *Esempio di matrice di dati*

i	ETA	PESO	ALTEZZA	SESSO	PROV
1	19	50	1.65	F	RM
2	22	75	1.78	M	RM
3	21	80	1.91	M	TO
4	23	56	1.72	F	NA
5	22	75	1.81	M	TO
6	20	58	1.68	F	NA

La prima volta che si usa questa funzione durante una sessione di lavoro si dovrà anche selezionare da una lista il sito *mirror* da cui scaricare il pacchetto.

In alternativa, nel menu "Pacchetti" si seleziona "Installa pacchetti" e, successivamente, si seleziona il pacchetto dalla lista di tutti i pacchetti.

2 Tipologie di dati in R

R lavora con dati strutturati. Le strutture di dati contemplate sono:

numeri 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 vettori multidimensionali. Le matrici hanno due dimensioni: righe e colonne. Gli *array* sono insiemi di numeri con p dimensioni. Ovviamente, per $p = 1$ si ottiene un vettore, per $p = 2$ una matrice. Il caso più generale, con $p > 2$ non verrà trattato;

data frame sono matrici bidimensionali dove ogni colonna può avere una tipo di dato diverso dalle altre;

liste insieme di elementi non omogenei tra loro.

Prima di andare a vedere i singoli tipi di dati, consideriamo il seguente esempio. Supponiamo di avere rilevato su sei unità, cinque variabili, tre numeriche, l'età, il peso e l'altezza, e due categoriche, il sesso e la provincia di nascita. I dati sono riportati nella Tabella 1. La prima colonna della matrice contiene le "etichette" con cui vengono identificate le unità. Ciascuna delle rimanenti colonne rappresenta una variabile osservata sul campione. La riga i-ma della Tabella 1 riporta i valori osservati per le cinque variabili sull'unità i.

In **R**, le variabili vengono rappresentate sotto forma di vettori. Ciascun vettore rappresenta una **distribuzione unitaria semplice**. La **matrice dei dati**, che contiene le colonne rappresentanti le variabili osservate sul campione è una **distribuzione unitaria multipla**.

Nel seguito, andremo a vedere dapprima come costruire i singoli vettori che compongono la Tabella 1 e, successivamente, la matrice dei dati. Nel far questo, verranno illustrate le principali proprietà di vettori e matrici di dati in **R**.

2.1 Vettori di dati

Per definire un **vettore** esistono molti modi, il più comune dei quali è la funzione `c()`. Per costruire i vettori numerici della matrice dei dati nella Tabella 1, scriveremo:

```
> ETA = c(19, 22, 21, 23, 22, 20)
> ETA

[1] 19 22 21 23 22 20

> PESO = c(50, 75, 80, 56, 75, 58)
> PESO

[1] 50 75 80 56 75 58

> ALTEZZA = c(1.65, 1.78, 1.91, 1.72, 1.81, 1.68)
> ALTEZZA

[1] 1.65 1.78 1.91 1.72 1.81 1.68
```

E' possibile effettuare operazioni sui vettori. Se, ad esempio, volessimo calcolare l'altezza in centimetri, piuttosto che in metri:

```
> ALT.CM = ALTEZZA * 100
> ALT.CM

[1] 165 178 191 172 181 168
```

Se, invece, vogliamo calcolare il *body mass index* delle unità del campione, dato dal rapporto tra il peso e il quadrato dell'altezza:

```
> BMI = PESO/ALTEZZA^2
> BMI

[1] 18.4 23.7 21.9 18.9 22.9 20.5
```

Per ridurre il numero di decimali, si utilizza la funzione `round(x, digits)`, dove `x` è un oggetto (numero, vettore, matrice) che contiene numeri da arrotondare, e `digits` è il numero di decimali.

```
> round(BMI, 2)

[1] 18.4 23.7 21.9 18.9 22.9 20.6
```

Per costruire i vettori delle variabili categoriche, si può ancora utilizzare la funzione `c`. Successivamente, si può applicare la la funzione `factor`, che definisce automaticamente le modalità della variabile categorica. Questo tipo di variabili sono utili nella classificazione delle unità. I singoli elementi del vettore vanno racchiusi tra virgolette: `" "`.

```
> SESSO = c("F", "M", "M", "F", "M", "F")
> SESSO

[1] "F" "M" "M" "F" "M" "F"

> SESSO = factor(SESSO)
> SESSO

[1] F M M F M F
Levels: F M

> PROV = c("RM", "RM", "TO", "NA", "TO", "NA")
> PROV = factor(PROV)
> PROV

[1] RM RM TO NA TO NA
Levels: NA RM TO
```

Per selezionare gli elementi di un vettore si deve inserire tra parentesi quadre, successivamente al nome del vettore, un indice, o un vettore di indici, che indichi la posizione degli elementi del vettore da estrarre. Se davanti all'indice c'è il segno `"-"`, il corrispondente elemento viene eliminato dal vettore:

```
> #seleziona il peso della prima unità
> PESO[1]

[1] 50

> #della sesta e della terza unità
> PESO[c(6, 3)]

[1] 58 80

> #elimina l'altezza della quarta unità
> ALTEZZA[- 4]

[1] 1.65 1.78 1.91 1.81 1.68

> #estrae l'età di chi ha un peso minore o uguale a 70 kg
> ETA[PESO <= 70]

[1] 19 23 20

> #estrae il peso di che ha un'altezza maggiore di 1.70
> PESO[ALTEZZA > 1.70]

[1] 75 80 56 75
```

```
> #estrae l'altezza di chi ha 22 anni
> ALTEZZA[ETA == 22]

[1] 1.78 1.81

> #estrae il peso delle unità di sesso femminile
> PESO[SESSO == "F"]

[1] 50 56 58

> #estrae l'altezza delle unità che hanno meno di 22 anni
> #e che non sono nate nella provincia di Napoli
> ALTEZZA[ETA < 22 & PROV != "NA"]

[1] 1.65 1.91
```

Osservazione Le variabili di classificazione possono essere, a loro volta trasformate in variabili numeriche, tramite la funzione `as.integer`. Si osservi che, nel caso i livelli siano definiti da caratteri, o stringhe di caratteri, la trasformazione in numeri rispetta l'ordine alfabetico delle componenti del vettore.

```
> as.integer(SESSO)

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

2.1.1 Tipologie particolari di vettori: le funzioni `seq` e `rep`

La funzione `seq` genera un vettore che contiene una sequenza regolare di numeri, in base a determinate regole. La sintassi della funzione è[3]:
`seq(from = 1, to, by, length)`

L'argomento `from`, che va sempre specificato, indica il primo numero della sequenza di numeri, che, di *default*, è posto pari ad 1. Il secondo, `to`, è il numero finale della sequenza. `by` indica il "passo" della sequenza, ossia l'incremento tra un numero e quello successivo. Se non specificato, è pari ad 1. Infine, `length` è la lunghezza della sequenza.

Non è necessario specificare tutti gli argomenti, dal momento che, ad esempio, una volta specificato valore di inizio, valore finale e passo, la lunghezza è automaticamente definita. Ad esempio, per ottenere una sequenza da 3 a 15, di passo 3 e, quindi, di lunghezza pari a 5, si può scrivere indifferentemente:

```
> #due diversi modi di ottenere la stessa sequenza
> x = seq(from = 3, to = 15, by = 3)
> x1 = seq(from = 3, to = 15, length = 5)
> x
```

[3] Si osservi che non sono stati riportati tutti gli argomenti della funzione, ma solo quelli di uso più comune. Questo verrà fatto anche per le funzioni che verranno successivamente illustrate, per non appesantire la spiegazione delle varie funzioni. Per approfondimenti, si può sempre fare ricorso all'*help*.

```
[1]  3  6  9 12 15
```

```
> x1
```

```
[1]  3  6  9 12 15
```

Si osservi anche che, come accennato precedentemente, non è necessario scrivere il nome dell'argomento, se viene riportato nell'ordine prefissato di *default*. Ad esempio, per ottenere un vettore con una sequenza di numeri da 2 (`from = 2`) a 10 (`to = 10`), di passo 2 (`by = 2`), è sufficiente scrivere:

```
> seq(2, 10, 2)
```

```
[1]  2  4  6  8 10
```

Se si specificano i nomi degli argomenti, questi possono essere posti anche in ordine diverso da quello prefissato:

```
> seq(1, length = 5, by = .5)
```

```
[1] 1.0 1.5 2.0 2.5 3.0
```

Infine, un modo compatto per ottenere delle sequenze di passo 1 è il seguente:

```
> 1:5
```

```
[1] 1 2 3 4 5
```

```
> 5:1
```

```
[1] 5 4 3 2 1
```

Nell'ultimo caso otteniamo una sequenza ordinata in maniera decrescente.

La funzione `rep(x, times)` genera un vettore in cui un oggetto (`x`), numero o vettore, viene ripetuto un numero di volte pari a (`times`):

```
> rep(x = 1, times = 5)
```

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

```
> rep(c(1, 2), 2)
```

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

2.2 *Data frame*

Le matrici in **R** possono essere definite tramite oggetti di "classe" `matrix` e `data.frame`. La differenza principale, è che gli oggetti di classe matrix sono matrici i cui elementi devono essere tutti della stessa natura, ad esempio vettori numerici, mentre gli oggetto di classe `data.frame` sono matrici in cui le colonne possono essere di natura differente.

Per questo motivo, questi ultimi oggetti sono impiegati per trattare **matrici di dati**. D'ora in avanti quando si parlerà di matrice di dati ci si riferirà ad un *data frame*, se non altrimenti specificato.

Si osservi, infine, che la maggior parte delle funzioni che possono essere applicate ad oggetti di classe `matrix` sono applicabili anche ai *data frame*, ma non è sempre vero il viceversa.

La funzione per generare un *data frame* è `data.frame`, che permette di unire più vettori di uguale lunghezza come colonne del *data frame*, ognuno dei quali si riferisce ad una diversa variabile.

Costruiamo quindi la matrice dei dati della Tabella 1:

```
> Dati = data.frame(ETA, PESO, ALTEZZA, SESSO, PROV)
> Dati

  ETA PESO ALTEZZA SESSO PROV
1  19   50    1.65     F   RM
2  22   75    1.78     M   RM
3  21   80    1.91     M   TO
4  23   56    1.72     F   NA
5  22   75    1.81     M   TO
6  20   58    1.68     F   NA
```

Si osservi che viene automaticamente generata la colonna delle etichette.

La funzione `str` permette di analizzare la struttura di un oggetto. In particolare, nel caso della matrice di dati appena ricavata si ha:

```
> str(Dati)

'data.frame':        6 obs. of  5 variables:
 $ ETA    : num  19 22 21 23 22 20
 $ PESO   : num  50 75 80 56 75 58
 $ ALTEZZA: num  1.65 1.78 1.91 1.72 1.81 1.68
 $ SESSO  : Factor w/ 2 levels "F","M": 1 2 2 1 2 1
 $ PROV   : Factor w/ 3 levels "NA","RM","TO": 2 2 3 1 3 1
```

La funzione `names` applicata alla matrice dei dati, restituisce il nome delle variabili della matrice:

```
> names(Dati)

[1] "ETA"     "PESO"    "ALTEZZA" "SESSO"   "PROV"
```

L'estrazione di dati da un *data frame* può essere effettuata in maniera simile a quanto avviene per i vettori. Va però tenuto conto che, trattandosi di una matrice, ha due dimensioni. Quindi, scrivendo `Dati[i, j]`, si estrae l'elemento che si trova nell'i-ma riga e nella j-ma colonna della matrice `A`. Quindi, l'indice prima della virgola si riferisce alle righe, mentre quello dopo la virgola riguarda le colonne:

```
> #estrae l'elemento che si trova nella prima riga e nella seconda colonna
> Dati[1, 2]

[1] 50

> #estrae il secondo ed il quarto elemento della prima riga
> Dati[1, c(2, 4)]

  PESO SESSO
1   50     F

> #estrae la sotto-matrice formata dalle prime due righe
> #e dalla seconda e terza colonna
> Dati[1:2, 2:4]

  PESO ALTEZZA SESSO
1   50    1.65     F
2   75    1.78     M
```

Si osservi che se non si scrive nulla prima della virgola, l'estrazione riguarderà tutte le righe; se non si scrive nulla dopo la virgola, lo stesso avverrà per le colonne. In altri termini, scrivendo `Dati[, j]` si estrarrà tutta la colonna j-ma, ossia tutte le righe della colonna; scrivendo `Dati[i, ]` si vuole estratte tutta la riga i-ma:

```
> #estrae la seconda riga
> Dati[2, ]

  ETA PESO ALTEZZA SESSO PROV
2  22   75    1.78     M   RM

> #estrae la terza colonna
> Dati[ , 3]

[1] 1.65 1.78 1.91 1.72 1.81 1.68

> #estrae la terza e quarta colonna
> Dati[ , 3:4]
```

```
  ALTEZZA SESSO
1    1.65     F
2    1.78     M
3    1.91     M
4    1.72     F
5    1.81     M
6    1.68     F
```

```
> #elimina la prima riga
> Dati[- 1, ]

  ETA PESO ALTEZZA SESSO PROV
2  22   75    1.78     M   RM
3  21   80    1.91     M   TO
4  23   56    1.72     F   NA
5  22   75    1.81     M   TO
6  20   58    1.68     F   NA
```

Dal momento che le colonne sono delle variabili, è possibile estrarle anche indicando nome della variabile, scrivendo `nomedata.frame$nomecolonna`. Ad esempio, per estrarre la prima variabile si può indicare l'indice della prima colonna, o il nome della variabile:

```
> Dati[, 1]

[1] 19 22 21 23 22 20

> Dati$ETA

[1] 19 22 21 23 22 20
```

Se quindi vogliamo ricavare la matrice dei dati contenente solo le variabili numeriche, scriveremo, ad esempio:

```
> Dati2 = Dati[, 1:3]
> Dati2

  ETA PESO ALTEZZA
1  19   50    1.65
2  22   75    1.78
3  21   80    1.91
4  23   56    1.72
5  22   75    1.81
6  20   58    1.68
```

2.2.1 Liste

Le **liste** sono insiemi di oggetti di natura e dimensione diversa. Per questo motivo, sono spesso usate per raccogliere i risultati di un'analisi. Per generare una lista si impiega la funzione `list`.

Costruiamo una lista, i cui elementi sono la nostra matrice dei dati, ed il vettore `x`, generato nell'esempio delle sequenze di numeri.

```
> Lista = list(Dati, x)
> Lista

[[1]]
  ETA PESO ALTEZZA SESSO PROV
1  19   50    1.65     F   RM
2  22   75    1.78     M   RM
3  21   80    1.91     M   TO
4  23   56    1.72     F   NA
5  22   75    1.81     M   TO
6  20   58    1.68     F   NA

[[2]]
[1]  3  6  9 12 15
```

Di *default* gli oggetti in una lista non sono identificati con un nome, ma solo con un numero. Per assegnare un nome a ciascun oggetto della lista si usa la funzione `names`, imputando un vettore di stringhe della stessa lunghezza del numero di oggetti[4]:

```
> names(Lista) = c("Dati", "sequenza")
> Lista

$Dati
  ETA PESO ALTEZZA SESSO PROV
1  19   50    1.65     F   RM
2  22   75    1.78     M   RM
3  21   80    1.91     M   TO
4  23   56    1.72     F   NA
5  22   75    1.81     M   TO
6  20   58    1.68     F   NA

$sequenza
[1]  3  6  9 12 15
```

Per estrarre un oggetto da una lista si può scrivere `nomelista[[numerooggetto]]`. Qualora ai diversi oggetti sia stato assegnato un nome, si può anche scrivere `nomelista$nomeoggetto`, . Quindi, per estrarre il secondo elemento della lista si può scrivere, alternativamente:

[4]La stessa funzione può essere impiegata per modificare i nomi degli oggetti di una lista, delle colonne di una matrice di dati.

```
> Lista[[2]]

[1]  3  6  9 12 15

> Lista$sequenza

[1]  3  6  9 12 15
```

2.3 Caricare dati da fonti esterne

E' raro che i dati vengano inseriti "a mano", come negli esempi precedenti. La situazione più comune è che vengano inseriti a partire da *file* esterni. **R** è in grado di leggere dati in praticamente qualsiasi formato[5].

In questa sezione vedremo come caricare in **R** dati che sono stati salvati sotto forma di testo, soluzione molto comune, con estensioni `.txt` o `.csv`. Per caricare questo tipo di dati si impiega la funzione `read.table()`, che, con le opportune specificazioni, è in grado di leggere *file* salvati con queste ed altre estensioni.

Consideriamo il *file* "exams.txt", che contiene i voti riportati in cinque materie da sei studenti. In questa matrice di dati le variabili sono gli esami e gli studenti sono le unità. Nella prima riga ci sono i nomi delle variabili, ossia gli esami sostenuti dagli studenti.

Perché **R** sia in grado di caricare questi dati, occorre indicare al programma dove si trova il *file*. Qualora il *file* si trovasse in un'altra cartella diversa dal *workspace* occorre indicare tutto il percorso, per permettere al programma di individuare la cartella. Ad esempio, se la cartella dove si trovano i dati fosse `"D:/my documents/data"`, si dovrebbe scrivere:
`read.table("D:/my documents/data/exams.txt", header = TRUE)`
dove l'argomento `header` posto uguale a `TRUE` sta a significare che la prima riga della matrice di dati contiene i nomi delle variabili.

Se, invece, la cartella in cui si trovano i dati è quella definita come spazio di lavoro, non serve indicarla. La soluzione più semplice è, quindi, copiare il *file* nella nostra cartella `"C:/lezioniR"`, in modo che il programma sia in grado di individuare i dati immediatamente, e scrivere:

```
> exams = read.table("exams.txt", header = TRUE)
> str(exams)

'data.frame':        6 obs. of  5 variables:
 $ EconomiaI   : int  18 30 23 21 30 27
 $ EconomiaII  : int  20 30 25 22 30 30
 $ StatisticaI : int  21 30 22 25 30 26
 $ StatisticaII: int  19 30 22 24 30 28
 $ Marketing   : int  26 30 28 28 26 24

> exams
```

[5]Tramite il pacchetto `foreign`, **R** è in grado di leggere e salvare dati in formati provenienti da altri programmi di analisi statistica, come Stata, SPSS e SAS.

```
        EconomiaI EconomiaII StatisticaI StatisticaII Marketing
Chiara         18         20          21           19        26
Piero          30         30          30           30        30
Sabrina        23         25          22           22        28
Elena          21         22          25           24        28
Sara           30         30          30           30        26
Marta          27         30          26           28        24
```

Una volta caricati i dati, si possono effettuare operazioni sulla matrice dei dati, o sulle singole variabili. Per accedere alle variabili della matrice dei dati si possono applicare le soluzioni viste precedentemente. Per cui, per richiamare la prima variabile si può scrivere, ad esempio:

```
> exams$EconomiaI

[1] 18 30 23 21 30 27
```

Se richiamassimo direttamente il nome della variabile, otterremmo un risultato di errore, perché **R** non può accedere direttamente alle variabili all'interno di una matrice di dati:

```
> EconomiaI
Error: object 'EconomiaI' not found
```

La funzione `attach()` permette di caricare temporaneamente in memoria le singole variabili della matrice dei dati e rappresenta, quindi, una maniera più veloce per accedere direttamente alle variabili della matrice dei dati:

```
> attach(exams)
> EconomiaI

[1] 18 30 23 21 30 27
```

Si osservi che comunque le variabili non compaiono nello spazio di lavoro:

```
> ls()

[1] "exams"
```

Osservazione Una volta effettuate le analisi sulle variabili, è preferibile applicare la funzione `detach()`, che "chiude" l'accesso diretto alle variabili.

Consideriamo ora il *file* "pazienti.csv", che riporta dati relativi a 15 persone, che sono state sottoposte ad un trattamento medico, sui quali sono state osservate le seguenti variabili[6]:

- `sesso`;
- `eta`;
- `col.pre`, colesterolo totale, prima del trattamento;
- `cor.pre`, cortisolo plasmatico, prima del trattamento;
- `umor.pre`, umore, prima del trattamento;
- `pess.pre`, pessimismo, prima del trattamento;
- `col.post`, colesterolo totale, dopo il trattamento;
- `cor.post`, cortisolo plasmatico, dopo il trattamento;
- `umor.post`, umore, dopo il trattamento;
- `pess.post`, pessimismo, dopo il trattamento;

I *file* con estensione `.csv` sono particolari tipi di documenti di testo che possono essere letti e salvati con Excel. In questo caso, il *file* è stato salvato con le impostazioni internazionali americane, per cui i decimali sono separati da un punto, e lo spazio tra le variabili è dato da una virgola[7]. Queste informazioni vanno date alla funzione `read.table()`, tramite gli argomenti `dec` e `sep` rispettivamente:

```
> pazienti = read.table("pazienti.csv", header = TRUE, dec = ".",
+ sep = ",", row.names = 1)
> head(pazienti)

  sesso eta peso col.pre cor.pre umor.pre pess.pre col.post cor.post umor.post
1     M  27   60     260    14.5        1        1      175      3.2         1
2     M  41   65     320     8.0        3        1      195      4.5         0
3     M  65   75     360    13.0        2        3      225      8.7         1
4     M  53   67     380    12.0        0        1      220      8.7         0
5     M  47   52     350    20.5        1        1      222     13.5         0
6     M  56   70     253    14.7        3        3      120     16.5         3
  pess.post
1         1
```

[6]Le variabili relative all'umore ed al pessimismo, sono state valutate in base alla scala della depressione di Beck.

[7]Se il *file* viene salvato con le impostazioni italiane, i decimali sono separati da una virgola, mentre le variabili da un punto e virgola.

```
2         0
3         3
4         1
5         0
6         3
```

L'argomento `row.names` posto uguale ad 1 indica al programma di non leggere la prima colonna del *data frame*, che contiene l'identificativo del paziente, come una variabile, ma di impiegarla per identificare le righe della matrice dei dati, ossia come etichette delle unità. La funzione `head` mostra solo le prime sei righe della matrice dei dati, il che è utile quando si ha a che fare con matrici molto grandi[8].

In alternativa, si può impiegare la funzione `read.csv()`, che è adatta a leggere dati salvati in `.csv`, con le impostazioni internazionali anglosassoni[9]:

```
> pazienti = read.csv("pazienti.csv", row.names = 1)
> str(pazienti)

'data.frame':        15 obs. of  11 variables:
 $ sesso    : Factor w/ 2 levels "F","M": 2 2 2 2 2 2 2 1 1 1 ...
 $ eta      : int  27 41 65 53 47 56 40 58 32 40 ...
 $ peso     : int  60 65 75 67 52 70 68 56 82 70 ...
 $ col.pre  : int  260 320 360 380 350 253 400 250 136 172 ...
 $ cor.pre  : num  14.5 8 13 12 20.5 14.7 7.2 13.3 13.6 8.3 ...
 $ umor.pre : int  1 3 2 0 1 3 1 3 2 2 ...
 $ pess.pre : int  1 1 3 1 1 3 0 3 2 3 ...
 $ col.post : int  175 195 225 220 222 120 245 320 165 148 ...
 $ cor.post : num  3.2 4.5 8.7 8.7 13.5 16.5 8.5 12.9 11.8 8 ...
 $ umor.post: int  1 0 1 0 0 3 0 3 0 0 ...
 $ pess.post: int  1 0 3 1 0 3 0 2 0 1 ...

> attach(pazienti)
```

Nella funzione `read.csv` non occorre specificare gli argomenti `header`, `dec` e `sep` che sono già impostati di *default*.

Osservazione Si noti che, sia per il *file* `exams` che per `pazienti`, abbiamo impiegato la funzione `attach()`, per cui non sarà necessario, nelle analisi che seguono, richiamare il nome della matrice dei dati di provenienza, ma solo i nomi delle variabili.

[8]Se si vuole visualizzare le prime n righe di una generica matrice `X`, si scrive: `head(X, n)`, dove `n` è compreso tra 1 e il numero di righe della matrice.

[9]L'equivalente per i *file* salvati con le impostazioni italiane è la funzione `read.csv2`.

3 Introduzione all'analisi dei dati

3.1 Analisi descrittiva dei dati

3.1.1 Le principali funzioni per l'analisi dei dati

In bf R ci sono diverse funzioni per l'analisi descrittiva dei dati. Tra le più utilizzate ricordiamo:

- `mean()` calcola la media di un vettore di dati;
- `median()` calcola la mediana di un vettore di dati;
- `var()` calcola la varianza di un vettore di dati, la covarianza tra due vettori, o la matrice di varianze e covarianze di una matrice di dati;
- `cor()` calcola la correlazione tra due vettori, o la matrice di correlazione di una matrice di dati;
- `sd()` calcola lo scarto quadratico medio (*standard error*) di un vettore di dati;
- `summary()` riporta le principali statistiche descrittive di un vettore o di una matrice di dati;
- `length()` restituisce la lunghezza del vettore (ossia la numerosità campionaria, se si opera su dati campionari);
- `sum()` calcola la somma degli elementi di un vettore.

Molte altre funzioni per l'analisi statistica dei dati saranno introdotte, di volta in volta, con esempi specifici. Alcuni esempi, basati sui dati del *file* `exams.txt` sono riportati di seguito. Per ridurre il numero di decimali, in maniera definita, senza dovere sempre utilzzare la funzione `round`, si utilizza la funzione `options`, che permette di modificare le opzioni di *default* di **R**. In particolare, scrivendo, ad esempio, `options(digits = 2)`, tutti i risultati non interi verranno riportati con due numeri decimali. Se dopo la virgola ci sono degli zero, riporterà i primi due numeri decimali dopo gli zeri[10]:

```
> #salva le impostazioni iniziali
> op = options()
> #riporta al massimo tre numeri decimali
> options(digits = 3)
> #La media può essere calcolata combinando le funzioni "sum" e "length"
> sum(EconomiaI)/length(EconomiaI)

[1] 24.8
```

[10] Dal momento che ogni modifica con `options` è definitiva, fino alla fine della sessione di lavoro, è opportuno salvare le impostazioni iniziali e ripristinarle alla fine dei calcoli.

```
> #o direttamente con la funzione "mean"
> mean(EconomiaI)

[1] 24.8

> #calcolo della mediana
> median(EconomiaI)

[1] 25

> #La funzione "var" può essere usata
> #per il calcolo della varianza di un vettore:
> var(EconomiaI)

[1] 24.6

> #per calcolare la covarianza tra due vettori:
> var(EconomiaI, EconomiaII)

[1] 21.6

> #o per ricavare la matrice di varianze e covarianze
> #tra le variabili di una matrice:
> var(exams)

            EconomiaI EconomiaII StatisticaI StatisticaII Marketing
EconomiaI        24.6       21.6        17.3         21.7       1.0
EconomiaII       21.6       20.2        14.1         18.9      -0.6
StatisticaI      17.3       14.1        14.7         16.8       1.6
StatisticaII     21.7       18.9        16.8         20.7       0.6
Marketing         1.0       -0.6         1.6          0.6       4.4

> #La funzione "cor" calcola la correlazione tra due variabili
> cor(EconomiaI, EconomiaII)

[1] 0.972

> #o la matrice di correlazione
> #tra le variabili di una matrice di dati quantitativi:
> cor(exams)

            EconomiaI EconomiaII StatisticaI StatisticaII Marketing
EconomiaI      1.0000     0.9719       0.913       0.9623    0.0962
EconomiaII     0.9719     1.0000       0.818       0.9250   -0.0637
StatisticaI    0.9132     0.8179       1.000       0.9642    0.1992
StatisticaII   0.9623     0.9250       0.964       1.0000    0.0629
Marketing      0.0962    -0.0637       0.199       0.0629    1.0000
```

```
> #calcolo dello scarto quadratico medio:
> sd(EconomiaI)

[1] 4.96

> #la funzione "summary", applicata ad un vettore numerico,
> #ne calcola valori minimo e massimo, media, mediana e quartili
> summary(EconomiaI)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   18.0    21.5    25.0    24.8    29.2    30.0

> #applicata ad una matrice,
> #riporta le principali informazioni per ciascuna variabile
> summary(exams)

   EconomiaI      EconomiaII    StatisticaI    StatisticaII    Marketing
 Min.   :18.0   Min.   :20.0   Min.   :21.0   Min.   :19.0   Min.   :24
 1st Qu.:21.5   1st Qu.:22.8   1st Qu.:22.8   1st Qu.:22.5   1st Qu.:26
 Median :25.0   Median :27.5   Median :25.5   Median :26.0   Median :27
 Mean   :24.8   Mean   :26.2   Mean   :25.7   Mean   :25.5   Mean   :27
 3rd Qu.:29.2   3rd Qu.:30.0   3rd Qu.:29.0   3rd Qu.:29.5   3rd Qu.:28
 Max.   :30.0   Max.   :30.0   Max.   :30.0   Max.   :30.0   Max.   :30
```

3.1.2 Le funzioni `apply`, `tapply` e `aggregate`

La funzione `apply` permette di calcolare una generica funzione sulle righe (qualora l'applicazione della funzione sulle righe abbia senso) o sulle colonne di una matrice di dati. La sintassi è `apply(X, dim, FUN)`, dove `X` è una matrice di dati, `dim` è la dimensione lungo la quale si vuole calcolare la funzione: 1 = per riga, 2 = per colonna), e `FUN` è la funzione da applicare ai dati[11].

```
> #calcola le medie delle variabili per riga
> apply(exams, 1, mean)

 Chiara   Piero Sabrina   Elena    Sara   Marta
   20.8    30.0    24.0    24.0    29.2    27.0

> #e per colonna
> apply(exams, 2, mean)

   EconomiaI   EconomiaII  StatisticaI StatisticaII    Marketing
        24.8         26.2         25.7         25.5         27.0
```

[11] Deve essere coerente con i dati che si vogliono analizzare. Non si può impiegare la funzione `mean`, ad esempio, se le variabili non sono quantitative.

In questo caso, dal momento che le variabili sono omogenee (sono voti riportati agli esami), ha senso sia la media per riga, voto medio dello studente, sia quella per colonna, voto medio per esame. Se le variabili non sono omogenee, non ha senso effettuare operazioni per riga, ma solo per colonna.

Se nel *data frame* sono presenti variabili di classificazione, è possibile calcolare una funzione separatamente su ciascuna classe di unità, tramite le funzioni `tapply` e `aggregate`, che si applicano, rispettivamente, ad un singolo vettore, e ad una matrice di dati.

La sintassi della funzione `tapply` è:
`tapply(X, INDEX, FUN)`
dove `X` è un vettore, `INDEX` una variabile di classificazione[12], o una lista di variabili di classificazione, e `FUN` la funzione da applicare ai dati.

Ad esempio, considerando la matrice di dati `pazienti`, se si vuole calcolare il peso medio dei pazienti, a seconda del sesso:

```
> pesomedio.s = tapply(peso, sesso, mean)
> pesomedio.s

   F    M
74.0 65.3
```

La sintassi della funzione `aggregate` è:
`aggregate(x, by, FUN)`
dove `x` è una matrice di dati, `by` è una lista, al limite composta da un solo oggetto, di variabili di classificazione, e `FUN`, la funzione.

Dalla matrice dei dati `pazienti` ricaviamo le variabili quantitative, e creiamo un nuovo *data frame*, sulle cui variabili calcoliamo la media in base al sesso:

```
> pazienti2 = data.frame(eta, peso, col.pre, cor.pre, col.post, cor.post)
> aggregate(pazienti2, list(sesso), mean)

  Group.1  eta peso col.pre cor.pre col.post cor.post
1       F 48.8 74.0     195    14.3      237    11.74
2       M 47.0 65.3     332    12.8      200     9.09
```

Il risultato è una matrice, sulle cui righe ci sono le medie delle variabili, distintamente per il sesso dei pazienti.

3.1.3 Distribuzioni di frequenze semplici e doppie

Per generare **distribuzioni di frequenze assolute** semplici (una sola variabile) o doppie (due variabili) in **R** si utilizza la funzione `table`. Anche se la funzione può essere applicata a qualsiasi tipo di variabile, ci occuperemo solo di distribuzioni di variabili qualitative, o quantitative discrete.

La distribuzione di frequenze della variabile `umor.pre`, ad esempio, è:

[12] Non necessariamente salvata come `factor`.

```
> fr = table(umor.pre)
> fr

umor.pre
0 1 2 3
1 4 7 3
```

La corrispondente distribuzione di frequenze relative si ottiene applicando la funzione `prop.table` alla distribuzione di frequenze assolute:

```
> prop.table(fr)

umor.pre
     0      1      2      3
0.0667 0.2667 0.4667 0.2000
```

Per determinare **distribuzioni di frequenza doppie**, si utilizza sempre `table`. Prendendo come esempio la distribuzione dei pazienti, in base al sesso ed alla valutazione dell'umore prima del trattamento:

```
> fr1 = table(sesso, umor.pre)
> fr1

     umor.pre
sesso 0 1 2 3
    F 0 1 6 1
    M 1 3 1 2

> prop.table(fr1) * 100

     umor.pre
sesso     0     1     2     3
    F  0.00  6.67 40.00  6.67
    M  6.67 20.00  6.67 13.33
```

dove la tabella delle frequenze relative è stata moltiplicata per 100, allo scopo di ottenere le percentuali, di più immediata leggibilità.

3.2 Creare grafici con R

R è in grado di produrre diversi tipi di grafico, attraverso funzioni per i grafici, che possono essere divisi in tre gruppi:

1. funzioni di alto livello, che creano un nuovo grafico sulla finestra grafica;
2. funzioni di basso livello, che aggiungono parti ad un grafico già esistente;

3. funzioni per grafici interattivi, che consentono di aggiungere interattivamente informazioni, o di estrarne, da un grafico esistente.

In **R** è inotre disponibile una lunga serie di parametri grafici, che permettono di personalizzare l'aspetto della finestra grafica, che possono essere modificati dall'utilizzatore. Si rimanda all'*help* in linea della funzione `par` per una lista di tali argomenti.

La funzione di alto livello principale e più generale è `plot()`, ma, a seconda dell'analisi condotta, si possono impiegare altre funzioni di alto livello, più specifiche. Ogni volta che si utilizza la funzione `plot()`, o qualsiasi altra funzione di alto livello, si apre una nuova finestra grafica, che sostituisce quella precedentemente aperta.

Lo *scatterplot*, o **diagramma di dispersione**, è un grafico cartesiano che permette di confrontare due variabili. E' formato dalle coordinate di ciascuna coppia di punti, che rappresentano il valore assunto da ciascuna unità nelle due variabili messe a confronto. Tramite il diagramma di dispersione è possibile evidenziare sia la presenza di valori anomali, ovvero punti che si discostano dalla nuvola dei punti, sia l'esistenza di relazioni tra le due variabili, ad esempio lineare o quadratica.

La funzione impiegata per lo *scatterplot* è `plot`. In questo caso, la sintassi della funzione è:

```
plot(x, y, type = "p")
```

dove `x` e `y` sono due vettori che rappresentano le coordinate dei punti nel grafico, mentre `type` definisce il tipo di grafico. Nel caso di un diagramma di dispersione, si usa l'opzione di *default*, ossia `type = "p"`, che equivale ad un grafico a punti.

A partire dai dati contenuti nel *file* `auto.csv` creiamo, come esempio, un grafico in cui si mettono a confronto il peso dell'auto (`peso`) e la cilindrata (`cilindrata`) (Figura 1):

```
> #carica i dati "auto"
> auto <- read.csv("auto.csv", row.names = 1)
> attach(auto)

> plot(x = peso, y = cilindrata)
```

In base al grafico si vede che c'è una correlazione positiva tra le due variabili.

Esercizio Calcolare la correlazione tra le due variabili, per verificare se l'intuizione grafica è giustificata.

Il grafico può essere arricchito, con altri argomenti della funzione `plot`, come, ad esempio, `main` per inserire un titolo al grafico, o `xlab` e `ylab` per inserire le intestazioni dell'asse delle ascisse e delle ordinate rispettivamente.

Possiamo anche modificare l'aspetto dei punti, con l'argomento `pch`, il cui valore di default è pari a 1, ed anche il colore, con l'argomento `col`, il cui valore di default è 1, equivalente a nero (Figura 2).

```
> plot(peso, cilindrata, xlab = "peso (kg)",
+ ylab = "cilindrata (l)", main = "Auto", pch = 2, col = 2)
```

Figure 1: *Scatterplot dei dati relativi a peso e cilindrata*

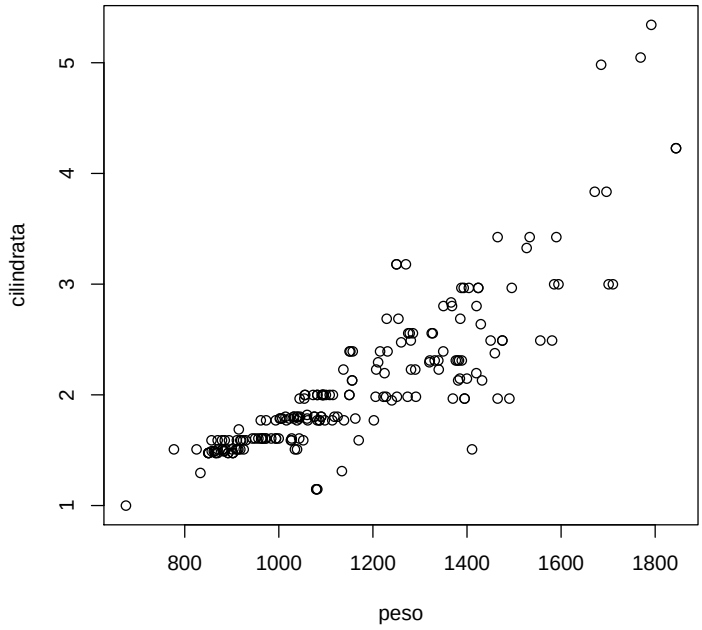


Figure 2: *Scatterplot dei dati relativi a peso e cilindrata con punti modificati*

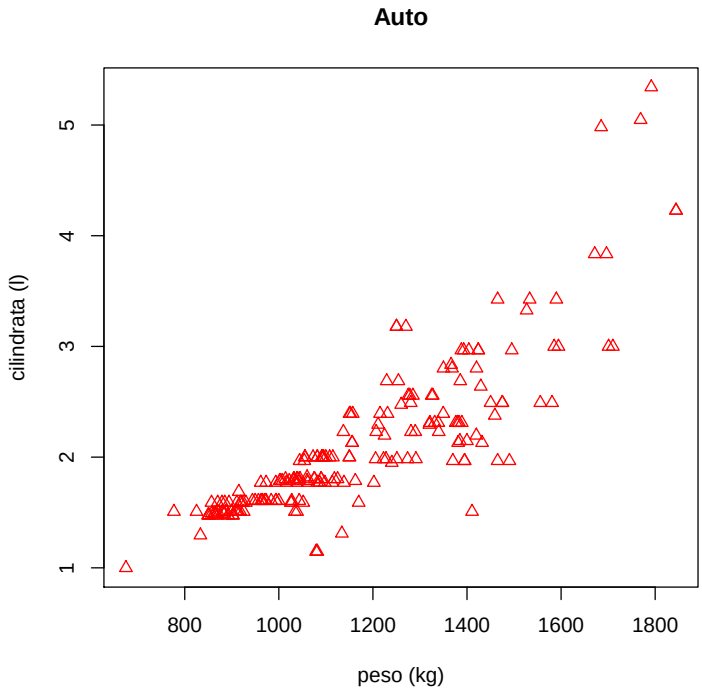
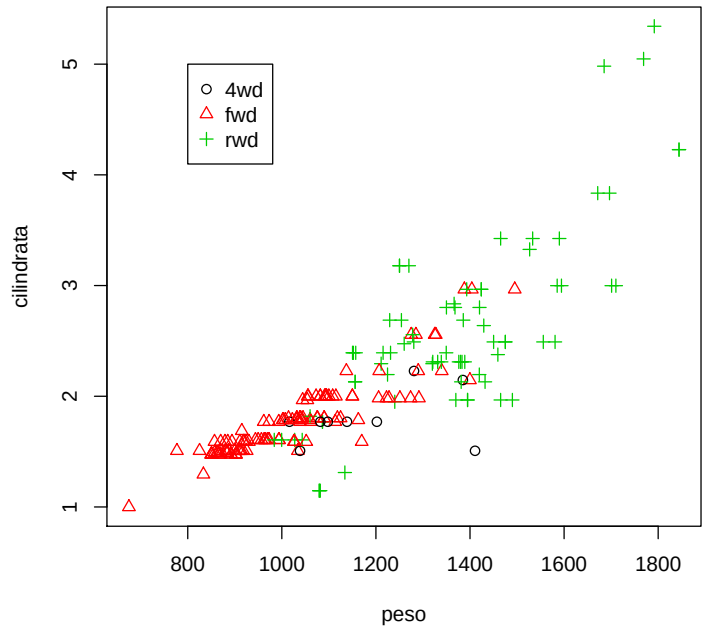


Figure 3: *Scatterplot dei dati relativi a peso e cilindrata con identificazione del tipo di trazione*



Esercizio Modificare `pch` e `col` per ottenere grafici con aspetto diverso.

Per identificare le unità sul grafico con un colore e/o con un punto diverso, a seconda del valore di una variabile di classificazione, o fattore, si possono impiegare nuovamente gli argomenti `pch` e/o `col`. In questo caso va specificato per i due argomenti un vettore numerico, che permetta di associare un tipo di punto e/o un colore a ciascuna unità, a seconda della classe cui appartiene. Se la variabile di classificazione è un vettore di stringhe, dovrà prima essere trasformata in vettore numerico tramite la funzione `as.integer`.

Consideriamo il caso in cui si voglia identificare le auto in base alla trazione. La variabile trazione non è numerica, quindi va trasformata. Dopo la trasformazione, si ha che `4wd = 1, fwd = 2, rwd = 3`.

Si può anche aggiungere una legenda, tramite la funzione `legend`, da inserire dopo la funzione `plot`, la cui sintassi di base è:
`legend(x, y, legend)`
dove `x` ed `y` sono le coordinate alle quali viene posizionata la legenda, mentre `legend` è un vettore, della stessa lunghezza del numero di classi diverse nella variabile di classificazione, che rappresenta i vari elementi da identificare. In questo caso si può impiegare direttamente la funzione `levels` per ricavare i nomi delle classi. In generale, si costruisce un vettore con i nomi delle modalità della variabile di classificazione. Altri argomenti variano a seconda del grafico. In questo caso occorrerà associare a ciascun elemento del vettore di stringhe il simbolo corrispondente, attraverso `pch` e `col` (Figura 3).

```
> traznum = as.integer(trazione)
```

```
> plot(peso, cilindrata, pch = traznum, col = traznum)
> legend(800, 5, legend = levels(trazione), pch = 1:3, col = 1:3)
```

Dal grafico si osserva che le automobili con quattro ruote motrici sono concentrate soprattutto tra quelle a bassa cilindrata e con un peso intorno alla media, le auto a trazione anteriore sono principalmente a bassa cilindrata e di piccolo peso, mentre tra quelle a trazione anteriore ci sono modelli con elevato peso e cilindrata.

Esercizio Creare un grafico simile al precedente, ma con `alimentazione` come variabile di classificazione. Commentare i risultati.

Il **grafico a bolle**, o *bubbleplot*, è un grafico bi-dimensionale che permette di rappresentare contemporaneamente l'associazione tra tre variabili, `x, y, z`. La terza variabile viene rappresentata sostituendo i punti corrispondenti alle coordinate delle unità rispetto alle prime due variabili con delle "bolle", la cui dimensione è proporzionale ai valori della terza variabile.

Per la generazione del *bubbleplot* in **R** si impiega la funzione `symbols`, la cui sintassi è:
`symbols(x, y, circles, inches, fg = 1, bg)`
dove `x, y` sono le coordinate delle prime due variabili, `circles` stabilisce la circonferenza delle bolle, proporzionali ai valori assunti dalla terza variabile, `inches` che fissa la massima ampiezza delle bolle, `fg` che specifica il colore del diametro della bolla (di *default* è nero = 1), mentre `bg` è un argomento che serve per determinare il colore con cui vanno riempite le bolle (di default non viene riempita). Altri argomenti sono comuni a quelli già visti per `plot`.

Consideriamo il caso in cui confrontiamo peso e cilindrata, con le bolle proporzionali alla percorrenza urbana (Figura 4):

```
> symbols(peso, cilindrata, circles = percorr.urbana,
+ inches = 0.5, xlab = "peso (kg)", ylab = "cilindrata (l)", main = "Auto")
```

In aggiunta a quanto visto con i precedenti grafici, ora osserviamo che, come prevedibile, i modelli più pesanti e con cilindrata più elevata hanno anche una percorrenza urbana minore.

Per distinguere le automobili in base all'alimentazione (`1 = benzina, 2 = diesel`), possiamo riempire ogni bolla con un colore diverso (Figura 5):

```
> symbols(peso, cilindrata, circles = percorr.urbana, inches = 0.5,
+ xlab = "peso (kg)", ylab = "cilindrata (l)",
+ bg = alimentazione, main = "Auto")
> legend(x = 880, y = 5,
+ legend = c("benzina", "diesel"), pch = rep(16, 2), col = 1:2)
```

In questo caso si mette in evidenza l'esistenza di due gruppi di auto con alimentazione diesel, un primo gruppo che si trova tra i modelli più al disotto dei 1200 kg, a bassa cilindrata e ad elevata percorrenza urbana, un secondo con cilindrata attorno alla media, piuttosto pesanti e con una percorrenza urbana medio-bassa.

Figure 4: *Bubble plot*

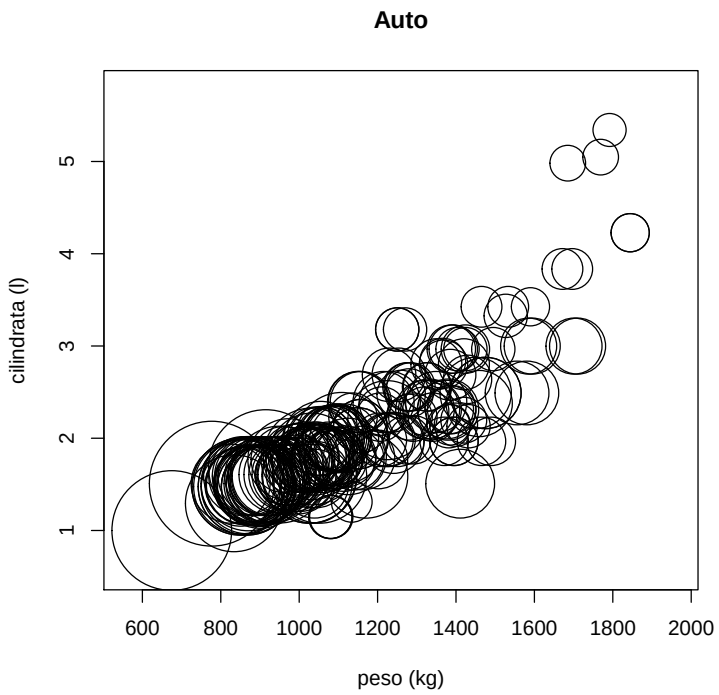


Figure 5: *Bubble plot con bolle distinte in base al tipo di alimentazione del motore*

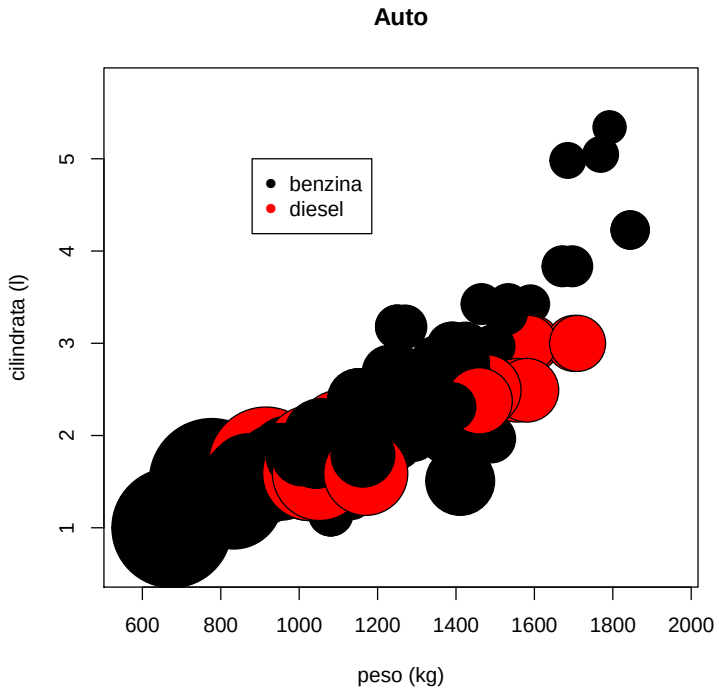
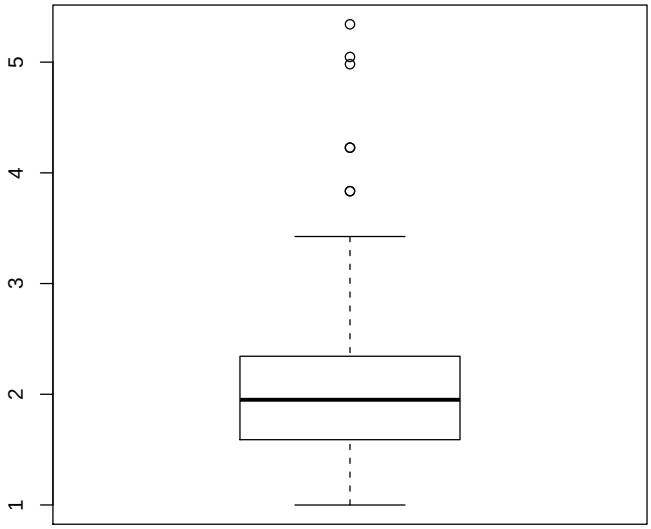


Figure 6: *Boxplot*



Esercizio Creare un grafico simile al precedente, ma con `trazione` come variabile di classificazione. Commentare i risultati.

Il *boxplot* è un grafico che rappresenta le seguenti quantità: minimo, massimo, primo, secondo (mediana) e terzo quartile. Fornisce informazioni sulla sulla variabilità che sulla asimmetria di una distribuzione. Si richiede con la funzione boxplot (Figura 6).

```
> boxplot(cilindrata)
```

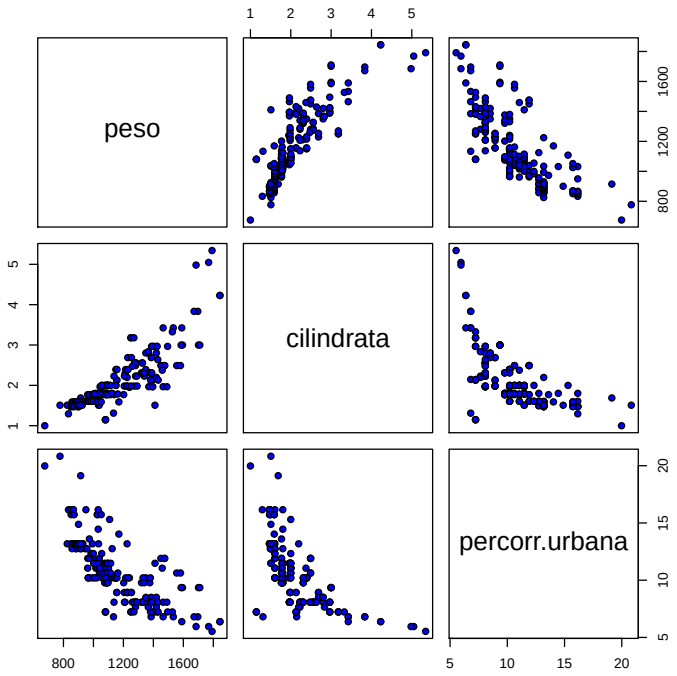
Un metodo per la rappresentazione contemporanea di tre o più variabili è la **matrice dei diagrammi di dispersione** (*scatterplot matrix*). Si tratta di un insieme di diagrammi di dispersione nei quali sono messe a confronto tutte le coppie di variabili di una matrice di dati quantitativi.

La funzione che genera la *scatterplot matrix* è `pairs`:
`pairs(x, labels)`
dove `x` è la matrice dei dati, mentre `labels` è il vettore dei nomi delle variabili. `labels` si utilizza, in maniera analoga alla funzione `names`, se si vuole modificare il nome delle variabili. Altri argomenti grafici sono identici a quelli visti per le precedenti funzioni.

Costruiamo la matrice dei diagrammi sul *data frame* `auto2`, ricavato dai dati `auto` (Figura 7):

```
> auto2 = data.frame(peso, cilindrata, percorr.urbana)
```

Figure 7: *Scatterplot matrix*




```
> pairs(x = auto2, pch = 21, bg = 4)
```

Guardando, ad esempio, alla prima riga della matrice dei grafici, si osserva una relazione positiva tra peso e cilindrata, mentre quella tra peso e percorrenza urbana è negativa.

Si possono distinguere le unità in base ad una variabile di classificazione, attraverso gli argomenti `pch` e `bg`, nello stesso modo visto negli esempi precedenti. In questo caso l'inserimento della legenda è molto più complicato e trascende l'obiettivo di questa dispensa.

Esercizio Creare un grafico in cui i punti e i colori dei punti cambino a seconda del tipo di trazione. Commentare i risultati.

Per variabili qualitative, o quantitative discrete, possiamo costruire un **diagramma a barre**, in cui l'altezza di ogni barra è proporzionale alla frequenza assoluta, o relativa, di ciascuna modalità della variabile, tramite la funzione `barplot`, applicata all'*output* di `table`.

Ad esempio, per la distribuzione di frequenze assolute dei modelli di auto in base al numero dei cilindri si ha (Figura 8:

```
> barplot(table(N.cilindri))
```

Infine, per variabili qualitative, non ordinate, si può utilizzare un **grafico a torta**, che consiste nell'attribuire ciascuna modalità della variabile ad una "fetta" di un cerchio, la cui ampiezza è proporzionale alla frequenza relativa della modalità, tramite funzione `pie`, applicata all'*output* di `table`

Con riferimento alla variabile `trazione` (Figura 9):

Figure 8: *Diagramma a barre*

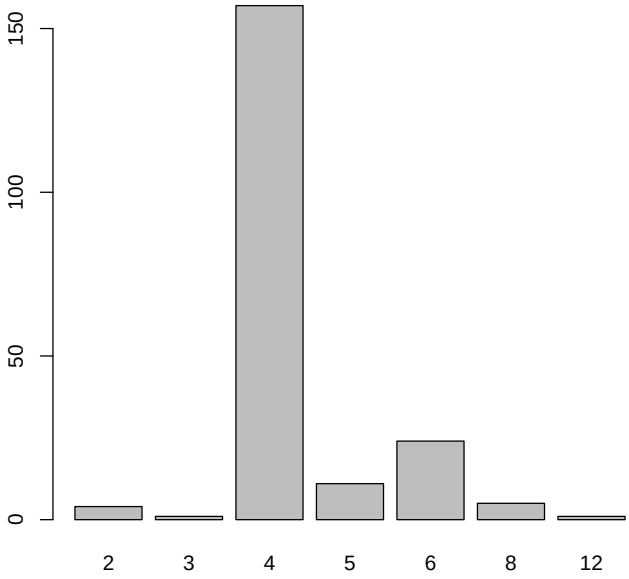
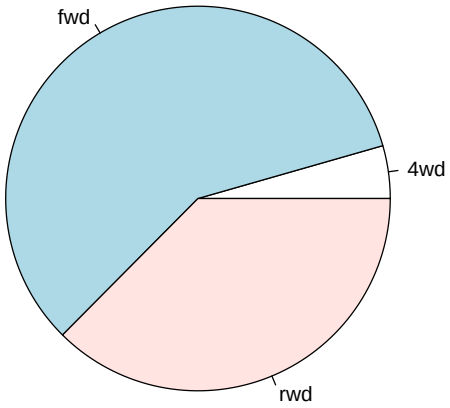


Figure 9: *Grafico a torta*



```
> pie(table(trazione))
```

4 Variabili casuali

Di seguito si riportano i comandi per il trattamento di variabili casuali (v.c.) in **R**.

Bernoulli

- Densità *dbinom*(x=x,size=1,prob=π)
- Ripartizione *pbinom*(q=x,size=1,prob=π)
- Quantile *qbinom*(p=α,size=1,prob=π)
- Estrazione di valori casuali da una v.c. Bernoulli *rbinom*(n,size=1,prob=π)

Binomiale

- Densità *dbinom*(x=x,size=m,prob=π)
- Ripartizione *pbinom*(q=x,size=m,prob=π)
- Quantile *qbinom*(p=α,size=m,prob=π)
- Estrazione di valori casuali da una v.c. binomiale *rbinom*(n,size=m,prob=π)

Geometrica

- Densità *dgeom*(x=x,prob=π)
- Ripartizione *pgeom*(q=x,prob=π)
- Quantile *qgeom*(p=α,prob=π)
- Estrazione di valori casuali da una v.c. geometrica *rgeom*(n,prob=π)

Poisson

- Densità *dpois*(x=x,lambda=λ)
- Ripartizione *ppois*(q=x,lambda=λ)
- Quantile *qpois*(p=α,lambda=λ)
- Estrazione di valori casuali da una v.c. Poisson *rpois*(n,lambda=λ)

Esponenziale

- Densità *dexp*(x=x,rate=λ)
- Ripartizione *pexp*(q=x,rate=λ)
- Quantile *qexp*(p=α,rate=λ)
- Estrazione di valori casuali da una v.c. esponenziale *rexp*(n,rate=λ)

Normale

- Densità *dnorm*(x=x,mean=μ,sd=σ)
- Ripartizione *pnorm*(q=x,mean=μ,sd=σ)
- Quantile *qnorm*(p=α,mean=μ,sd=σ)
- Estrazione di valori casuali da una v.c. normale *rnorm*(n,mean=μ,sd=σ)

Student

- Densità *dt*(x=x,df=k)
- Ripartizione *pt*(q=x,df=k)
- Quantile *qt*(p=α,df=k)
- Estrazione di valori casuali da una v.c. t di Student *rt*(n,df=k)

Chi-quadrato

- Densità *dchisq*(x=x,df=k)
- Ripartizione *pchisq*(q=x,df=k)
- Quantile *qchisq*(p=α,df=k)
- Estrazione di valori casuali da una v.c. χ^2 *rchisq*(n,df=k)

Fisher

- Densità *df*(x=x,df1=n1,df2=n2)
- Ripartizione *pf*(q=x,df1=n1,df2=n2)
- Quantile *qf*(p=α,df1=n1,df2=n2)
- Estrazione di valori casuali da una v.c. F di Fisher *rf*(n,df1=n1,df2=n2)

Uniforme

- Densità *dunif*(x=x,min,max)
- Ripartizione *punif*(x=x,min,max)
- Quantile *qunif*(p=α,min,max)
- Estrazione di valori casuali da una v.c. F di Fisher *runif*(n,min,max)

4.1 Esempi

Probabilità di ottenere un valore pari a 3 (3 successi) in una binomiale n=10 prove indipendenti π=0.5.

```
> dbinom(3,10,0.5)

[1] 0.117
```

Funzione di ripartizione valore 3 (3 successi) in una binomiale n=10 prove indipendenti π=0.5.

```
> pbinom(3,10,0.5)

[1] 0.172
```

Quantile 0.171875 in una binomiale in una binomiale n=10 prove indipendenti π=0.5.

```
> qbinom(0.171875,10,0.5)

[1] 3
```

Generazione casuale di 5 valori in una binomiale n=10 prove indipendenti π=0.5.

```
> rbinom(5,10,0.5)

[1] 7 6 5 6 3
```

Vettore k contenente i numeri da 0 a 10

```
> k<-c(0:10)
> k

 [1]  0  1  2  3  4  5  6  7  8  9 10
```

Vettore contenente le probabilità dei valori da 0 a 10 in una binomiale n=10 prove indipendenti π=0.5 (Figura 10).

```
> p<-dbinom(k,10,0.5)
> p

 [1] 0.000977 0.009766 0.043945 0.117188 0.205078 0.246094 0.205078 0.117188
 [9] 0.043945 0.009766 0.000977
```

Distribuzione di probabilità della funzione di probabilità dei valori da 0 a 10 in una binomiale n=10 prove indipendenti π=0.8 (Figura 11).

```
> p<-dbinom(k,10,0.8)
```

Figure 10: *Grafico a barre verticali della funzione di probabilità dei valori da 0 a 10 in una binomiale n=10 prove indipendenti π=0.5*

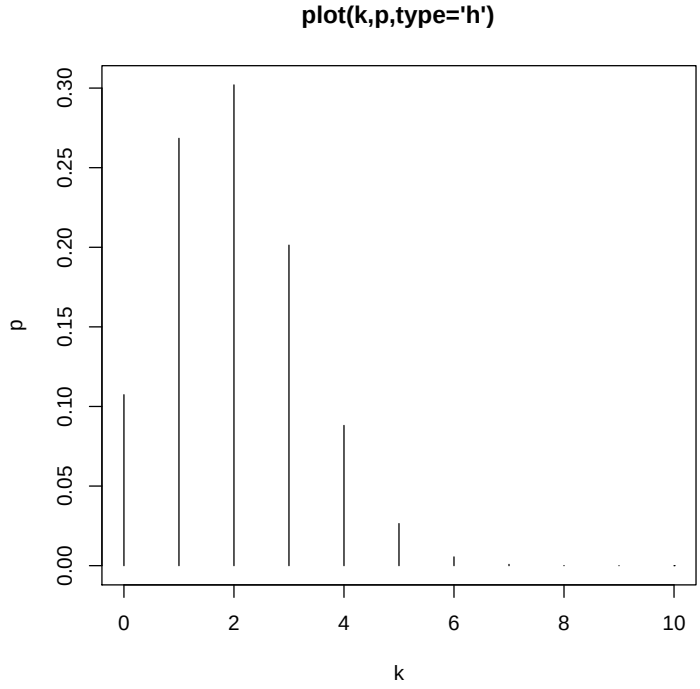


Figure 11: *Grafico a barre verticali della funzione di probabilità dei valori da 0 a 10 in una binomiale n=10 prove indipendenti π=0.8*

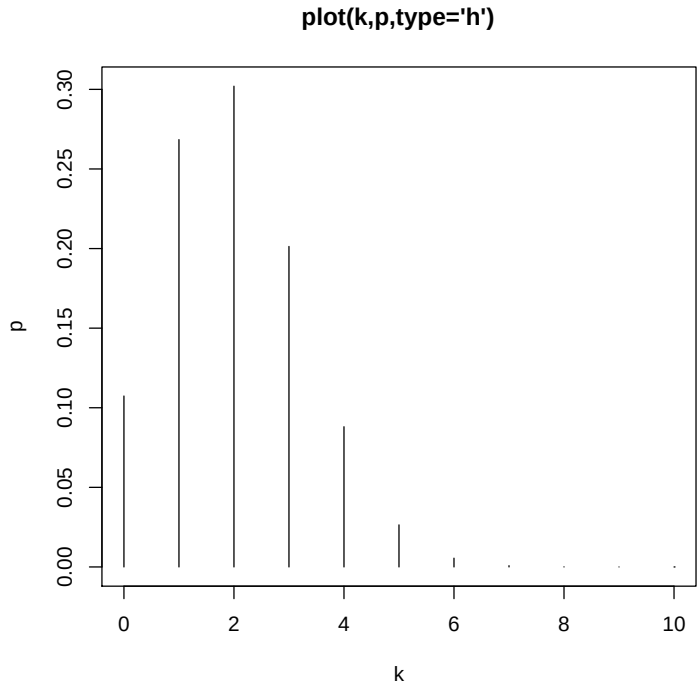


Figure 12: *Grafico a barre verticali della funzione di probabilità dei valori da 0 a 10 in una binomiale n=10 prove indipendenti π=0.2*

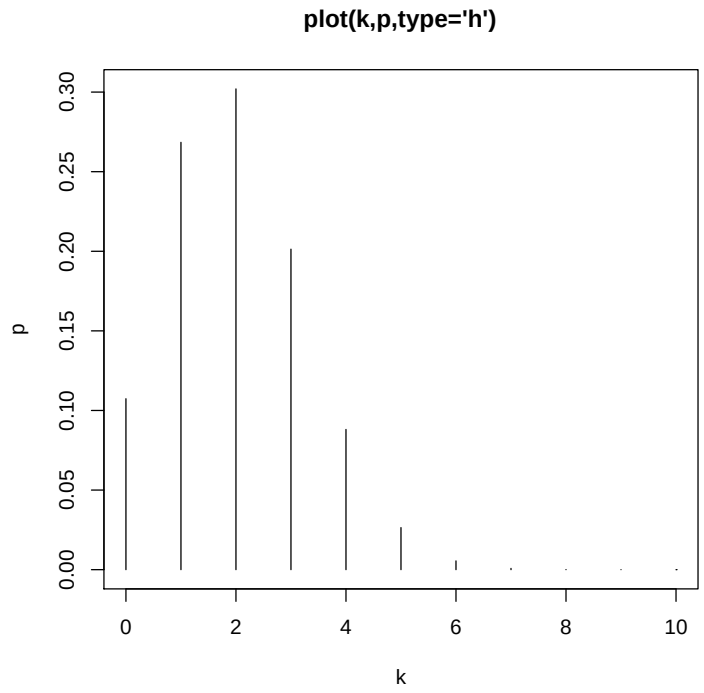


Figure 13: *Grafico della distribuzione Normale con media uguale e diversa varianza o varianza uguale e diversa media*

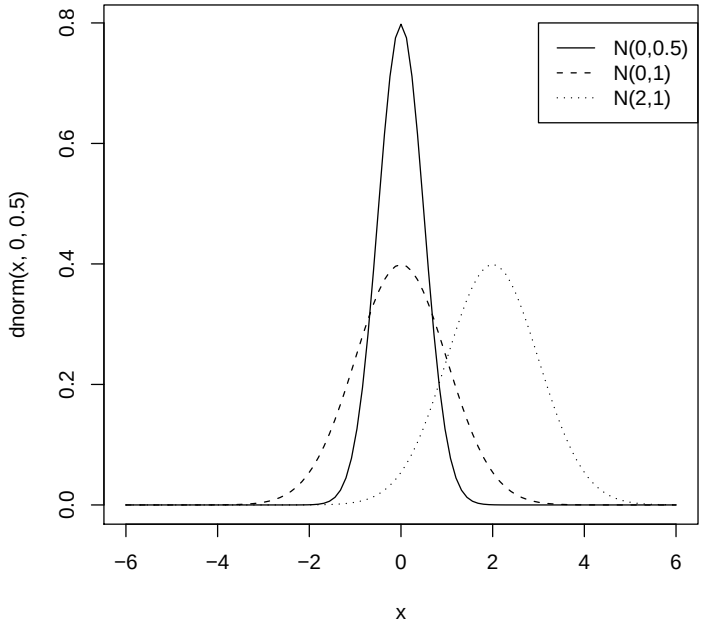
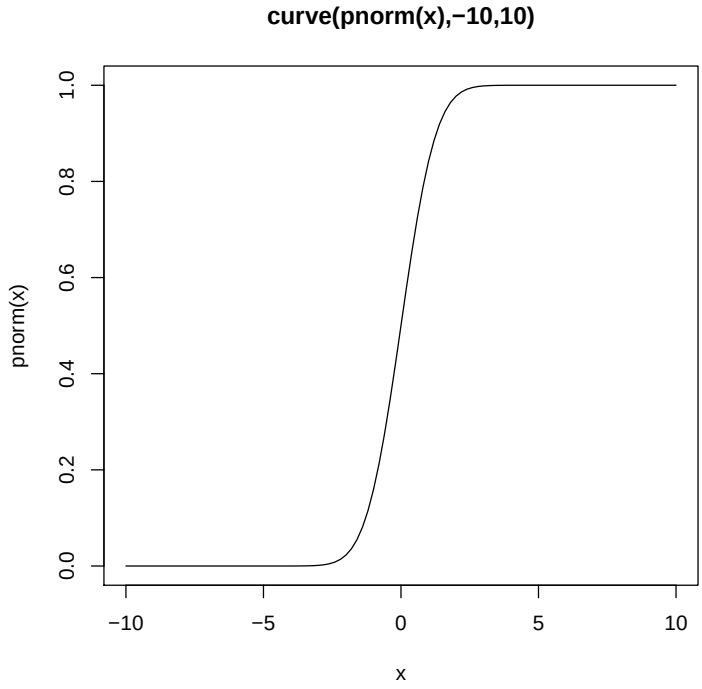


Figure 14: *Grafico funzione di ripartizione Normale standard*



Distribuzione di probabilità della funzione di probabilità dei valori da 0 a 10 in una binomiale n=10 prove indipendenti π=0.2 (Figura 12).

```
> p<-dbinom(k,10,0.2)
```

Quantile 0.95 di una Normale standard

```
> qnorm(0.95)

[1] 1.64
```

Quantile 0.975 di una Normale standard

```
> qnorm(0.975)

[1] 1.96
```

Quantile 0.99 di una Normale standard

```
> qnorm(0.99)

[1] 2.33
```

Quantile 0.025 di una Normale standard

```
> qnorm(0.025)

[1] -1.96
```

Quantile 0.975 di una t Student di parametro 10

```
> qt(0.975,10)

[1] 2.23
```

Quantile 0.975 di una t Student di parametro 20

```
> qt(0.975,20)

[1] 2.09
```

Quantile 0.95 di una chi-quadro di parametro 5

```
> qchisq(0.95,5)

[1] 11.1
```

Quantile 0.95 di una F di Fisher di parametri 2 e 3

```
> qf(0.95,2,3)

[1] 9.55
```

Il grafico della distribuzione Normale con media uguale e diversa varianza o varianza uguale e diversa media si ottiene con la seguente sintassi (Figura 13):

```
> curve(dnorm(x,0,0.5),-6,6)
> curve(dnorm(x,0,1),-6,6,add=TRUE)
> curve(dnorm(x,2,1),-6,6,add=TRUE)
```

5 Verifica delle ipotesi

5.1 Test di ipotesi sul valor medio

Dato un campione estratto da una variabile casuale X, con distribuzione normale di media μ e varianza incognita σ^2, se si vuole sottoporre a verifica l'ipotesi che la media del campione sia pari a μ_0, la verifica dell'ipotesi viene effettuata sulla statistica test:

$$t = \frac{\overline{X} - \mu_0}{s/\sqrt{n}},$$

dove $\overline{X}$ è la media campionaria, s è la stima dello scarto quadratico medio e n è la numerosità del campione. Sotto l'ipotesi nulla, la statistica t si distribuisce come una variabile casuale T di Student, con $n-1$ gradi di libertà.

Per il test di ipotesi sul valore medio di un campione, in **R** si impiega la funzione `t.test`, che permette:

1. di costruire un intervallo di confidenza per la media;
2. di eseguire test di ipotesi sia su un singolo campione, sia per il confronto tra due campioni.

La sintassi di base della funzione è:

```
t.test(x, y = NULL, alternative = "two.sided", mu = 0,
paired = FALSE, var.equal = FALSE, conf.level = 0.95)
```

dove `x` è un vettore di dati, `y` è un vettore (opzionale) di dati, da specificare quando si vuole effettuare il confronto tra due campioni, mentre `mu` è l'ipotesi nulla sul valore della media (test su un campione), o sulla differenza tra medie (test su due campioni). Di *default* `mu = 0`. L'argomento `alternative` è una stringa che serve a specificare l'ipotesi alternativa. Può essere `"two.sided"` (opzione di *default*), quando l'ipotesi alternativa è che la media (o la differenza tra medie) è diversa da quella fissata nell'ipotesi nulla, `"less"`, quando è minore, `"greater"`, quando è maggiore. Infine `conf.level` è il livello di confidenza dell'intervallo, ovvero il complemento ad 1 della probabilità dell'errore di I specie fissato, di *default* posto uguale a 0.95. Gli argomenti `paired` e `var.equal` verranno commentati successivamente.

L'*output* della funzione è una lista che contiene, tra gli altri, i seguenti oggetti:

`statistic` il valore della statistica t;

`parameter` i gradi di libertà della statistica t;

`p.value` il *p-value* del test, che è la probabilità di ottenere un risultato pari (o più estremo) di quello osservato, supposta vera l'ipotesi nulla, $Pr(|T| > t)$;

`conf.int` l'intervallo di confidenza per la media;

`estimate` il valore stimato della media, o della differenza tra medie;

`null.value` il valore specificato per l'ipotesi nulla;

Eseguendo il test senza salvare il risultato, o richiamando il nome con cui vengono salvati i risultati, **R** restituisce le principali informazioni circa il test condotto.

5.1.1 Test *t* sulla media di un campione

A partire dalle due variabili del *dataframe* `pazienti` che riportano il livello di colesterolo prima e dopo il trattamento, creiamo una nuova variabile, `ratio`, data dal rapporto tra le due. Questa variabile è un numero indice, minore di 1 se il livello di cortisolo è diminuito dopo il trattamento, e maggiore se vale il viceversa.

```
> ratio = cor.post/cor.pre
> summary(ratio)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.221   0.611   0.745   0.805   0.985   1.470
```

L'ipotesi nulla che si vuole sottoporre a verifica è che non ci sia differenza nel livello di colesterolo, contro l'ipotesi alternativa che ci sia differenza. In termini formali:

$$\begin{aligned} H0: &\quad \mu = 1 \\ H1: &\quad \mu \neq 1 \end{aligned}$$

Fissando un livello di significatività pari a $\alpha = 0.05$, si ha:

```
> t1 = t.test(ratio, mu = 1)
> t1

        One Sample t-test

data:  ratio
t = -2.33, df = 14, p-value = 0.03535
alternative hypothesis: true mean is not equal to 1
95 percent confidence interval:
 0.625 0.985
sample estimates:
mean of x
    0.805

> t1$statistic

    t
-2.33

> t1$p.value

[1] 0.0353

> t1$estimate

mean of x
    0.805
```

Nell'esempio abbiamo evidenziato sia le informazioni generali sul test, sia alcune delle informazioni più rilevanti. Ricordando che il *p-value* può anche essere interpretato come il minimo livello di significatività per il quale l'ipotesi nulla viene rifiutata, dal momento che è minore di α, l'ipotesi nulla viene rifiutata.

Un altro modo di valutare i risultati del test è quello di confrontare la statistica t con il valore critico, o soglia, della distribuzione T di Student, al livello di significatività prefissato. Per ricavare i valori della distribuzione si impiega la funzione `qt(p, df)`, dove `p` è una probabilità, o un vettore di probabilità, mentre `df` è il numero di gradi di libertà. In pratica, per un dato valore di p, la funzione restituisce il valore t_p tale che $Prob(T_{df} \leq t_p) = p$, dove T_{df} è una variabile casuale T di Student, con df gradi di libertà.

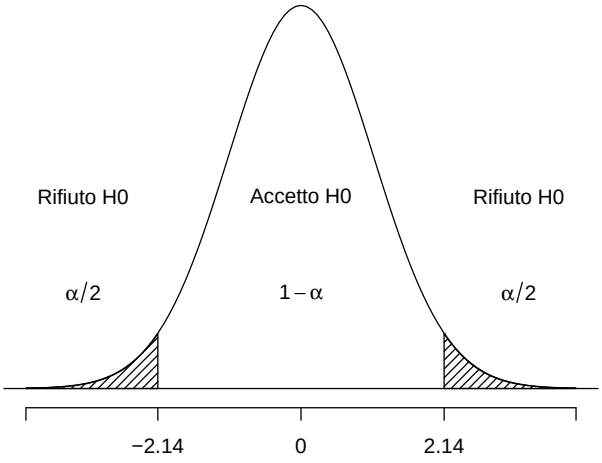
Dal momento che il test è bi-direzionale, ossia l'ipotesi nulla può essere rifiutata sia per valori positivi che negativi della statistica t, ricaviamo i due valori critici corrispondenti a $t_{\alpha/2}$ e a $t_{1-\alpha/2}$.

```
> n = length(ratio)
> n

[1] 15

> alpha = .05
> qt(c(alpha/2, 1 - (alpha/2)), n - 1)

[1] -2.14  2.14
```

La distribuzione della statistica, con le regioni di accettazioni e di rifiuto, è illustrata nella Figura[13] 15.

Il valore osservato per la statistica t non è compreso all'interno dei due valori critici, per cui, come visto precedentemente, l'ipotesi nulla viene rifiutata.

Figure 15: *Distribuzione della statistica* t
Test bidirezionale



In realtà, l'ipotesi che si vorrebbe verificare è se il livello di cortisolo post-trattamento sia significativamente minore di quello pre-trattamento. In tal senso, si potrebbe effettuare un test unidirezionale, del tipo:

$$\begin{aligned} H0: &\quad \mu = 1 \\ H1: &\quad \mu < 1 \end{aligned}$$

[13] La sintassi per ottenere la funzione è piuttosto complicata, e non verrà commentata. E' tuttavia disponibile nello *script* della dispensa, per chi vuole approfondire.

dove l'ipotesi nulla è, al solito, che non ci sia differenza, contro l'ipotesi alternativa che, in media, il livello di cortisolo post-trattamento sia inferiore a quello osservato prima del trattamento.

La distribuzione della statistica test, con la regione di accettazione e quella di rifiuto è riportata in Figura 16.

Per effettuare il test in questo caso si scriverà:

```
> t.test(ratio, mu = 1, alternative = "less")

        One Sample t-test

data:  ratio
t = -2.33, df = 14, p-value = 0.01767
alternative hypothesis: true mean is less than 1
95 percent confidence interval:
  -Inf 0.952
sample estimates:
mean of x
    0.805
```

Anche in questo caso l'ipotesi nulla viene rifiutata.

Esercizio Calcolare il valore critico per il test unidirezionale.

5.1.2 Test t per campioni indipendenti e appaiati

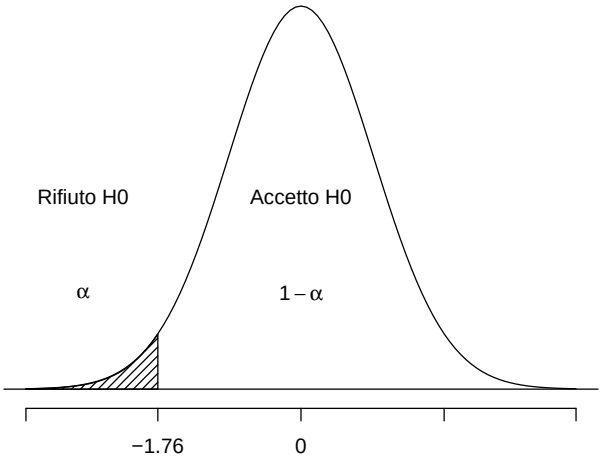
Il test può essere condotto anche per confrontare le medie di due campioni. Consideriamo dapprima il caso di campioni indipendenti.

Sempre con riferimento agli stessi dati dell'esempio precedente, vogliamo verificare se il peso dei pazienti di sesso femminile sia maggiore di quello dei pazienti di sesso maschile. Costruiamo quindi due nuove variabili:

```
> peso.m = peso[sesso == "M"]
> peso.f = peso[sesso == "F"]
> mean(peso.m)

[1] 65.3

> mean(peso.f)

[1] 74
```

$$\begin{aligned} H0: &\quad \mu_F = \mu_M \\ H1: &\quad \mu_F > \mu_M \end{aligned}$$

Figure 16: *Distribuzione della statistica* t
Test unidirezionale



Si osservi che l'ipotesi nulla corrisponde a $\mu_F - \mu_M = 0$, che equivale al valore di *default* per `mu` nella funzione `t.test`. Facciamo inoltre l'ulteriore ipotesi che le due varianze incognite siano uguali[14]. In questo caso, si utilizza un argomento opzionale della funzione, ossia `var.equal` che va posto uguale a `TRUE`. Infine, fissiamo la probabilità dell'errore di I specie pari ad $\alpha = 0.01$:

```
> t.test(peso.f, peso.m, var.equal = TRUE,
+ conf.level = 0.99, alternative = "greater")

        Two Sample t-test

data:  peso.f and peso.m
t = 1.49, df = 13, p-value = 0.08
alternative hypothesis: true difference in means is greater than 0
99 percent confidence interval:
 -6.78   Inf
sample estimates:
mean of x mean of y
     74.0      65.3
```

L'ipotesi nulla viene accettata.

[14] In realtà, questa ipotesi può essere sottoposta a verifica, tramite la funzione `var.test`.

Osservazione Nel caso in cui le varianze delle due popolazioni non possano essere assunte uguali, si pone `var.equal = FALSE`.

Un'altra situazione comune è quella dei campioni appaiati. L'esempio più comune è proprio quello relativo ai nostri dati, ossia verificare l'efficacia del trattamento, confrontando il valore medio di una qualche variabile del campione prima e dopo l'esperimento. Per questo tipo di test si utilizza l'argomento `paired`, con l'opzione tt TRUE.

Per questo tipo di test consideriamo un nuovo *data frame*, `sleep`[15], che contiene dati relativi all'effetto di due diversi sonniferi, in termini di incremento di numero di ore di sonno, su 10 pazienti. Le variabili sono:

`extra` : numero extra di ore di sonno, rispetto ad un gruppo di controllo;

`group` : tipo di sonnifero.

In base al numero medio di ore di sonno extra nei due gruppi si osserva che, in media, con il secondo tipo di sonnifero i pazienti guadagnano in media più ore di sonno. Si vuole verificare se questa differenza è significativa.

```
> attach(sleep)
> tapply(extra, group, mean)

   1    2
0.75 2.33

> extra1 = extra[group == 1]
> extra2 = extra[group == 2]
> t.test(extra2, extra1, alternative = "greater", paired = T)

        Paired t-test

data:  extra2 and extra1
t = 4.06, df = 9, p-value = 0.001416
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 0.867   Inf
sample estimates:
mean of the differences
                  1.58
```

Come ci si aspetterebbe, l'ipotesi nulla, che corrisponde ad assumere che non ci sia differenza tra i due gruppi viene rifiutata, a favore dell'ipotesi alternativa che la media del secondo gruppo è significativamente maggiore di quella del primo.

[15] Nella *release* di **R** sono inclusi un gran numero di *data frame*, che possono essere richiamati semplicemente specificando il nome del *data frame*. Scrivendo `data()`, comparirà una finestra con l'intera lista dei dati disponibili.

Esercizio Il test sulle medie di campioni appaiati equivale ad un test sulle differenze tra i due campioni. Verificare per esercizio.

5.2 Anova univariata

L'analisi della varianza (ANOVA) permette di confrontare due o più gruppi di unità, definiti in base ad una o più variabili di classificazione, tramite il confronto tra la variabilità interna di questi gruppi con la variabilità tra i gruppi. L'ipotesi nulla è che tutti i gruppi abbiano la stessa media, e che le differenze osservate tra i gruppi siano dovute solo al caso.

La funzione che si impiega è `anova`, che viene applicata al risultato di una regressione lineare di una variabile continua su una o più variabile discrete[16]. Nel caso in cui si consideri una sola variabile esplicativa, si parla di analisi della varianza univariata, o ad una via.

Consideriamo di nuovo i dati relativi alla matrice `pazienti`.

L'ipotesi nulla che vogliamo verificare è che, a prescindere dal livello di autovalutazione del pessimismo dopo il trattamento, il livello di colesterolo, sempre dopo il trattamento, sia sempre lo stesso, contro l'ipotesi alternativa che almeno due gruppi siano differenti tra loro in media.

In **R** bisogna prima effettuare la regressione della variazione del livello di cortisolo sull'umore dopo il trattamento, e poi effettuare l'analisi della varianza. La regressione lineare tra le due variabili si ottiene con la funzione `lm`[17]:

```
> fit = lm(col.post ~ factor(pess.post))
> fit

Call:
lm(formula = col.post ~ factor(pess.post))

Coefficients:
       (Intercept)  factor(pess.post)1  factor(pess.post)2  factor(pess.post)3
             216.7               -11.0                88.3               -44.2

> anova.fit = anova(fit)
> as.matrix(anova.fit)

                  Df Sum Sq Mean Sq F value Pr(>F)
factor(pess.post)  3  19849    6616    3.12 0.0701
Residuals         11  23313    2119      NA     NA
```

Si osservi che la variabile `umor.post` è stata trasformata in una variabile di classificazione, tramite la funzione `factor`.

Prima di commentare i risultati, commentiamo l'*output* delle due funzioni. La funzione `lm` restituisce i coefficienti stimati della regressione[18]. Nell'analisi della varianza il valore di

[16] Le variabili esplicative devono essere salvate come `factor`.

[17] La tilde tra le due variabili nella funzione `lm` si ottiene tenendo premuto il tasto "Alt" e digitando 126.

[18] In realtà, la maggior parte del'*output*, come spesso avviene in **R** non è immediatamente visibile, ma è salvata in una lista, in questo caso salvata con il nome `fit`.

tali coefficienti non è di particolare interesse, quanto piuttosto il loro segno. Si noti, intanto, che sono riportati i coefficienti di tutti i gruppi, tranne il primo, `"A"`. Questo perché **R** lo considera come gruppo di riferimento. In effetti, l'intercetta è proprio il numero medio di insetti trovati vivi negli esperimenti nei quali si usa questo tipo di spray. I rimanenti valori dei coefficienti rappresentano il valore differenziale rispetto al gruppo di riferimento.

L'*output* della funzione `anova`, che per comodità è stato riportato come una matrice, tramite la funzione `as.matrix`, restituisce la scomposizione della varianza e della devianza sia nei gruppi che tra i gruppi.

Nelle prime tre colonne della prima riga della matrice ci sono i valori relativi alla devianza tra i gruppi. Il primo valore è il numero di gradi di libertà (pari al numero di gruppi, meno 1), il secondo è la devianza tra i gruppi e il terzo la varianza. La seconda riga riporta i corrispondenti valori per la devianza nei gruppi.

Tornando alla prima riga, il quarto valore è il valore della statistica F che si utilizza per sottoporre a verifica l'ipotesi che le medie dei diversi gruppi sono uguali. Si osservi che il valore della statistica è pari al rapporto tra la varianza tra i gruppi e quella nei gruppi. L'idea è che se le medie sono uguali tra loro, la varianza dei gruppi è nulla, per cui tale rapporto è pari a 0. In generale, per accettare l'ipotesi nulla ci si aspettano valori piccoli di tale rapporto. L'ultimo valore è il corrispondente *p-value*, ossia la probabilità di osservare un valore maggiore di quello osservato per la statistica F. Se il valore della statistica è sufficientemente piccolo, il *p-value* sarà elevato, e l'ipotesi nulla non può essere rifiutata[19].

Nel nostro caso, il test risulta non significativo, perché il *p-value* è piccolo, ma comunque superiore ai livelli di significatività generalmente fissati, per cui si accetta l'ipotesi nulla.

5.3 Test χ^2

In presenza di una tabella di contingenza è possibile calcolare il valore di χ^2, per verificare la presenza di connessione o di indipendenza tra le due variabili della tabella. Il test χ^2 serve a verificare se il valore della statistica è significativamente diverso da 0, nel qual caso c'è dipendenza tra le due variabili.

Costruiamo la tabella di contingenza per le variabili `sesso` e `umor.post`. Si vuole verificare l'ipotesi nulla che ci sia indipendenza tra le due variabili.

```
> tab = table(sesso, umor.post)
> tab

     umor.post
sesso 0 1 2 3
    F 5 0 2 1
    M 4 2 0 1
```

Per effettuare il test, si utilizza la funzione `summary`, applicata alla tabella:

```
> summary(tab)
```

[19] Nella seconda riga questi valori non possono essere calcolati, e il programma restituisce un valore di dato mancante, `NA`.

```
Number of cases in table: 15
Number of factors: 2
Test for independence of all factors:
        Chisq = 4, df = 3, p-value = 0.3
        Chi-squared approximation may be incorrect

> #ripristina le impostazioni iniziali di R
> options(op)
```

In alternativa si può utilizzare la funzione `chisq.test`.

Il valore del *p-value* associato a questo test è superiore agli usuali livelli di significatività che vengono fissati, per cui l'ipotesi di indipendenza non può essere rifiutata.