Introduzione a R

Claudio Agostinelli [1]
Dipartimento di Statistica
Università di Padova
Padova

Versione 0.3

Ottobre 2000

[1] Claudio Agostinelli, Dipartimento di Statistica, Via San Francesco, 33, Università di Padova, Padova, e-mail: claudio@stat.unipd.it

Prefazione

Questa raccolta di materiale, è stata scritta, in più volte, nell'ambito delle esercitazioni svolte per il Corso di Statistica (Corso di base) per il Diploma in Statistica ed Informatica per la Gestione delle Imprese dell'Università di Padova e del Corso di Statistica per il Diploma in Statistica ed Informatica per la Gestione delle Imprese dell'Università di Venezia, sede di Treviso. L'attuale versione, presenta gli elementi essenziali del linguaggio e introduce alcune semplici analisi statistiche, in particolare utilizzando rappresentazioni grafiche e modelli di regressione. Per quest'ultimo argomento lo stesso esempio è ripreso più volte, utilizzando R come "calcolatrice" o invece utilizzando le funzioni proprie di R . Lungi dall'essere in una veste definitiva, esso viene reso pubblico nella speranza che possa essere utile.

Sarò grato a chiunque segnali errori e imprecisioni.

Gli insiemi di dati e i file di codice sono disponibili presso:

http://www.stat.unipd.it/˜ claudio

Padova, 20 Ottobre 2000

Claudio Agostinelli

Indice

Elenco delle figure

Parte I

Introduzione al linguaggio di R

Capitolo 1

Introduzione

R è un programma per la statistica e la grafica. Esso consiste principalmente di un linguaggio e di un insieme di librerie che consentono di svolgere la maggior parte delle analisi statistiche necessarie per la descrizione di un fenomeno. Il presente manuale è stato scritto per rendere più agevole l'introduzione al linguaggio di R .

Per entrare in R dalla finestra principale di *Windows95-98* posizionarsi con il *mouse* sull'icona corrispondente ed eseguire un doppio "click" con il tasto sinistro. Sullo schermo apparirà una finestra (*console*) riconoscibile per la presenza del *prompt* > da cui si potranno scrivere i comandi desiderati, altri finestre potranno essere aperte per ospitare i grafici. Per gli utenti di *Linux* è sufficiente digitare il comando `R` da un terminale (per la grafica è necessario essere in un ambiente X). Per vedere quali sono gli oggetti che sono stati creati fino ad ora si possono utilizzare i comandi `objects()` o `ls()`.

Per avere informazioni su di un comando è necessario utilizzare un *browers*, ad esempio *Netscape* e caricare il *file* in formato HTML opportuno, questo potrà avvenire usando il comando `help.start()`, oppure con, `help(`nomecomando`)` otterremo informazioni sullo specifico comando direttamente sulla *console*.

Infine per concludere una sessione di lavoro digitiamo il comando `q()`. Prima di uscire il programma ci chiederà se vogliamo salvare la sessione di lavoro. Questo ci consente di riprendere il lavoro dal punto in cui lo avevamo lasciato.

Tutti i comandi sono funzioni del tipo `nomecomando()`. Se si omettono le parentesi R stamperà il contenuto della funzione invece di eseguirla.

Per le operazioni di somma, sottrazione, moltiplicazione e divisione si utilizzano gli usuali simboli: + - * /. Per l'elevazione a potenza si usa: ** oppure ^ .

Ad esempio:

```
> 18*2
[1] 36
>
```

L'*output* viene stampato sul video preceduto da un indice contenuto tra parentesi quadre. Vedremo in seguito quale sia la sua utilità.

È necessario prestare attenzione all'uso delle parentesi, ad esempio:

```
> 2+3*4
[1] 14
>
```

non è equivalente a

```
> (2+3)*4
[1] 20
>
```

Così come

```
> 4*3^2
[1] 36
>
```

non equivale a

```
> (4*3)^2
[1] 144
>
```

Bisogna quindi tener presente che l'elevamento a potenza precede la moltiplicazione (e la divisione) che a sua volta precede l'addizione (e la sottrazione).

Gli operatori logici corrispondono invece ai seguenti simboli:

$<$	minore di
$>$	maggiore di
$<=$	minore o uguale a
$>=$	maggiore o uguale a
$==$	uguale a
$!=$	diverso da
&	AND
$\|$	OR
!	NOT

Quindi si ottiene:

```
> 4!=4
[1] FALSE
>
```

Che indica che l'affermazione è falsa. Come falsa risulta anche l'uguaglianza:

```
> (4*3)^2==4*3^2
[1] FALSE
>
```

mentre invece:

```
> (4*3)^2==4^2*3^2
[1] TRUE
>
```

Da notare che risulta:

```
> TRUE==1
[1] TRUE
> FALSE==0
[1] TRUE
>
```

Per assegnare il valore 21 alla variabile `x` si digita:

```
> x<-21
>
```

Digitando semplicemente il nome della variabile si può vedere il contenuto:

```
> x
[1] 21
>
```

mentre con il comando `rm()` possiamo rimuovere il contenuto di una variabile

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

Alternativamente, per assegnare un valore alla variabile `x` si può usare la seguente sintassi:

```
> x<-21
>
```

R è sensibile ai caratteri maiuscoli e minuscoli così le variabili `x` e `X` non sono la stessa cosa. È possibile creare variabili contenenti più valori (cioè vettori) con il comando `c()`:

```
> vett<-c(1,2,3,4)
> vett
[1] 1 2 3 4
>
```

Abbiamo quindi creato un vettore di nome `vett` di quattro elementi.

Attenzione a non usare per il nome di una variabile i nomi riservati a funzioni di R .

È possibile estrarre valori dai vettori ponendo l'indice di posizione dell'elemento nel vettore tra parentesi quadre. Ad esempio:

```
> vett[2]
[1] 2
> vett[2:4]
[1] 2 3 4
> vett[c(1,3)]
[1] 1 3
> vett[-c(2,3)]
[1] 1 4
>
```

L'ultimo esempio mostra come sia possibile utilizzare il segno di sottrazione per escludere valori.

L'estrazione di elementi da un vettore può essere effettuato altresì con gli operatori logici. Costruiamo, ad esempio, un nuovo vettore:

```
> vett<-c(100,18,41,21,53)
>
```

Se vogliamo ad esempio i valori più grandi di 50:

```
> vett>50
[1]  TRUE FALSE FALSE FALSE  TRUE
> vett[vett>50]
[1] 100  53
>
```

Se vogliamo i valori compresi tra 20 e 80:

```
> vett[vett>20 & vett<80]
[1] 41 21 53
>
```

Se moltiplichiamo un vettore per uno scalare si ottiene che ogni elemento del vettore viene moltiplicato per lo scalare:

```
> vett*3
[1] 300  54 123  63 159
>
```

Cosìper tutte le altre operazioni. In generale quindi le operazioni eseguite sul vettore vengono effettuate su ciascuno dei sui elementi. Se costruiamo un altro vettore:

```
> vett2<-c(27,28,29,30,31)
>
```

e lo sommiamo al vettore `vett` si ottiene la somma elemento per elemento (i vettori devono avere lo stesso numero di elementi).

```
> vett+vett2
[1] 127  46  70  51  84
>
```

analogamente per le altre operazioni.

Possiamo aggiungere elementi ad un vettore. Ad esempio possiamo unire il vettore `vett` e il vettore `vett2` in un unico vettore che chiamiamo `vett1`:

```
> vett1<-vett+vett2
> vett1
[1] 127  46  70  51  84
>
```

La funzione `Length()` fornisce il numero di elementi di un vettore. La funzione `sum` consente invece di ottenere la somma degli elementi contenuti in un vettore:

```
> sum(vett)
[1] 233
>
```

La funzione `cumsum()` consente invece di avere la somma cumulata:

```
> cumsum(vett)
[1] 100 118 159 180 233
>
```

La media del vettore `vett` può essere semplicemente ottenuta nel seguente modo:

```
> sum(vett)/length(vett)
[1] 46.6
>
```

Esiste comunque la funzione `mean()` che fornisce direttamente la media aritmetica:

```
> mean(vett)
[1] 46.6
>
```

Per ordinare i valori si può utilizzare la funzione `sort()`

```
> svett<-sort(vett)
> svett
[1]  18  21  41  53 100
>
```

Dato che la dimensione di `vett` è dispari allora la mediana è il valore che occupa la posizione $5/2+1=3$, quindi:

```
> svett[5/2+1]
[1] 41
>
```

Oppure più semplicemente usando la funzione `median()`:

```
> median(vett)
[1] 41
>
```

La mediana può essere ottenuta anche dalla funzione `quantile()` che calcola i quantili:

```
> quantile(vett,probs=0.5)
50%
 41
>
```

Dove `probs=0.5` indica che si desidera il quantile corrispondente al livello 0.5. Cosìper avere il primo e terzo quartile:

```
> quantile(vett,probs=c(0.25,0.75))
25% 75%
 21  53
>
```

Per calcolare lo scarto interquartile si può sfruttare la funzione `diff()`:

```
> diff(quantile(vett,probs=c(0.25,0.75)))
75%
 32
>
```

Infine la funzione `summary()` fornisce con un unico comando vari indicatori di sintesi:

```
> summary(vett)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   18.0    21.0    41.0    46.6    53.0   100.0
>
```

Per il calcolo della varianza possiamo utilizzare la definizione:

```
> sum((vett-mean(vett))^2)/length(vett)
[1] 879.44
>
```

mentre la funzione `var()` fornisce la varianza campionaria:

```
> var(vett)
[1] 1099.3
> var(vett)*(length(vett)-1)/length(vett)
[1] 879.44
>
```

Si ricorda che il denominatore della varianza campionaria è pari alla numerosità meno 1.

Alcuni riferimenti sono:

Venables, B., Smith, D., Gentleman, R. e Ihaka, R. (1997), *Notes on* R : A Programming Environment for Data Analysis and Graphics, disponibile in ogni CRAN in formato PostScript.

Everitt, B.S. (1994), *A Handbook of Statistical Analyses using S-plus*, Chapman & Hall, New York.

Krause, A. e Olson, M. (1997), *The Basic of S and S-plus*, Springer, New York.

Venables, W.N. e Ripley, B.D., *Modern Applied Statistics with S-plus*, Springer Verlag, New York.

È possibile ottenere il programma e altre informazioni in ogni CRAN, ai seguenti indirizzi:

http://cran.r-project.org/	(Austria)
http://cran.at.r-project.org/)	(Austria)
http://cran.dk.r-project.org/	(Danimarca)
http://cran.it.r-project.org/	(Italia)
http://cran.ch.r-project.org/	(Svizzera)
http://cran.uk.r-project.org/	(Gran Bretagna)
http://cran.us.r-project.org/	(Stati Uniti)

Capitolo 2

Uso delle matrici

Vediamo come costruire un vettore con il comando matrix:

```
> matrix(1:6)
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
[5,]    5
[6,]    6
```

Abbiamo creato un vettore colonna.

Per costruire una matrice è necessario specificare il numero di righe e/o il numero di colonne, così per costruire una matrice con due righe usiamo il comando:

```
> matrix(1:6,nrow=2)
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Come si vede, la matrice viene riempita colonna per colonna e il numero di colonne è stato calcolato come il numero di elementi diviso il numero di righe. Per riempire la matrice riga per riga usiamo l'opzione byrow:

```
> matrix(1:6,nrow=2,byrow=TRUE)
     [,1] [,2] [,3]
```

```
[1,]    1    2    3
[2,]    4    5    6
```

Costruiamo ora il vettore x:

```
> x<-3:8
> x
[1] 3 4 5 6 7 8
```

nel seguito mostriamo alcuni modi per costruire matrici, si osservi i diversi risultati:

```
> matrix(x,3,2)
     [,1] [,2]
[1,]    3    6
[2,]    4    7
[3,]    5    8
> matrix(x,ncol=2)
     [,1] [,2]
[1,]    3    6
[2,]    4    7
[3,]    5    8
> matrix(x,ncol=3)
     [,1] [,2] [,3]
[1,]    3    5    7
[2,]    4    6    8
> matrix(x,ncol=3,byrow=TRUE)
     [,1] [,2] [,3]
[1,]    3    4    5
[2,]    6    7    8
```

Passiamo ad analizzare un esempio un pò più complesso. Costruiamo con il comando c il vettore dati che contiene nell'ordine il PIL (Prodotto Interno Lordo), la Popolazione e l'Inflazione di tre nazioni della comunità Europea.

```
> dati<-c(197,8,1.8,1355,58,1.7,2075,81,1.8)
> dati
[1]  197.0    8.0    1.8 1355.0   58.0    1.7 2075.0   81.0    1.8
```

Con il comando matrix costruiamo la matrice nazioni.dati:

```
> nazioni.dati<-matrix(dati,nrow=3,byrow=TRUE)
> nazioni.dati
     [,1] [,2] [,3]
[1,]  197    8  1.8
[2,] 1355   58  1.7
[3,] 2075   81  1.8
```

Non è facile da questa matrice ricordare in quale riga si trovino le nazioni e in quale colonna le variabili. Per facilitare questa operazione possiamo usare il comando dimnames. Il dimnames della mareice nazioni.dati è vuoto:

```
> dimnames(nazioni.dati)
NULL
> dim(nazioni.dati)
[1] 3 3
```

e la dimensione della nostra matrice è 3×3. Costruiamo due vettori, uno contenente i nomi delle nazioni e l'altro il nome delle variabili:

```
> nazioni<-c("Austria","Francia","Germania")
> nazioni
[1] "Austria"  "Francia"  "Germania"
> variabili<-c("PIL","Pop","Inflazione")
> variabili
[1] "PIL"        "Pop"        "Inflazione"
```

Con il comando che segue assegnamo il nome alle righe (il comando list è un comando che serve per creare liste, oggetti che sono molto più generali delle matrici):

```
> dimnames(nazioni.dati)<-list(nazioni,NULL)
> nazioni.dati
         [,1] [,2] [,3]
 Austria  197    8  1.8
 Francia 1355   58  1.7
Germania 2075   81  1.8
```

In questa maniera assegnamo il nome alle colonne:

```
> dimnames(nazioni.dati)<-list(NULL,variabili)
> nazioni.dati
```

```
      PIL Pop Inflazione
[1,]  197   8        1.8
[2,] 1355  58        1.7
[3,] 2075  81        1.8
```

E come segue assegnamo il nome alle righe e alle colonne:

```
> dimnames(nazioni.dati)<-list(nazioni,variabili)
> nazioni.dati
          PIL Pop Inflazione
 Austria  197   8        1.8
 Francia 1355  58        1.7
Germania 2075  81        1.8
```

Adesso i nomi delle righe e delle colonne della matrice nazioni.dati non sono vuoti:

```
> dimnames(nazioni.dati)
[[1]]:
[1] "Austria"  "Francia"  "Germania"

[[2]]:
[1] "PIL"        "Pop"        "Inflazione"
```

Vediamo ora come accedere agli elementi di una matrice. Per accedere all'elemento che è nella prima riga e nella seconda colonna scriveremo:

```
> nazioni.dati[1,2]
[1] 8
```

se invece vogliamo avere tutta la prima riga scriveremo:

```
> nazioni.dati[2,1:3]
  PIL Pop Inflazione
 1355  58        1.7
```

o in maniera più semplice ed efficacie:

```
> nazioni.dati[2,]
  PIL Pop Inflazione
 1355  58        1.7
```

Possiamo individuare un elemento di una matrice anche riferendoci ai nomi come negli esempi che seguono:

```
> nazioni.dati["Germania","Inflazione"]
[1] 1.8
> nazioni.dati[,"Inflazione"]
 Austria Francia Germania
     1.8     1.7      1.8
```

Per modificare il valore di un elemento è sufficiente assegnare all'elemento il nuovo valore:

```
> nazioni.dati["Austria","Pop"]
[1] 8
> nazioni.dati["Austria","Pop"]<-10
> nazioni.dati["Austria","Pop"]
[1] 10
> nazioni.dati["Austria","Pop"]<-8
```

Una delle cose più interessanti di R è che può svolgere operazioni tra le matrici, create ad esempio le due matrici A e B:

```
> A<-matrix(0:5,2,3)
> A
     [,1] [,2] [,3]
[1,]    0    2    4
[2,]    1    3    5
> B<-matrix(seq(0,10,2),2,3)
> B
     [,1] [,2] [,3]
[1,]    0    4    8
[2,]    2    6   10
```

possiamo calcolare la matrice che contiene le somme elemento per elemento scrivendo semplicemente:

```
> A+B
     [,1] [,2] [,3]
[1,]    0    6   12
[2,]    3    9   15
```

oppure possiamo sommare lo scalare d a tutti gli elementi della matrice A:

```
> d<-5
> d
[1] 5
> d+A
     [,1] [,2] [,3]
[1,]    5    7    9
[2,]    6    8   10
```

Se vogliamo invece moltiplicare la matrice A per la B usando la moltiplicazione delle matrici (moltiplicazione righe per colonne) allora usiamo il comando:

```
> A%*%B
Error in A %*% B : non-conformable arguments
>
```

ma attenzione che le dimensioni delle matrici siano corrette, in particolare il numero di colonne di A deve essere uguale al numero di righe di B. Per moltiplicare A e B possiamo in questo caso trasporre la matrice B con il comando t:

```
> C<-t(B)
> C
     [,1] [,2]
[1,]    0    2
[2,]    4    6
[3,]    8   10
```

e rieseguire la moltiplicazione:

```
> A%*%C
     [,1] [,2]
[1,]   40   52
[2,]   52   70
```

Fate attenzione che l'operatore * compie solo una moltiplicazione elemento per elemento e così pure l'operatore di divisione /:

```
> A*B
     [,1] [,2] [,3]
[1,]    0    8   32
[2,]    2   18   50
> A/B
     [,1] [,2] [,3]
[1,]   NA  0.5  0.5
[2,]  0.5  0.5  0.5
```

Se vogliamo ottenere l'inversa di una matrice quadrata non singolare usiamo il comando `solve()`:

```
> D<-A%*%C
> solve(D)
           [,1]       [,2]
[1,]  0.7291667 -0.5416667
[2,] -0.5416667  0.4166667
```

Le operazioni possono essere svolte tra vettori della stessa matrice:

```
> nazioni.dati[,"PIL"]/nazioni.dati[,"Pop"]
 Austria  Francia Germania
  24.625 23.36207 25.61728
```

il risultato in questo caso è il Prodotto Interno Lordo pro capite. Se vogliamo aggiungere una nuova colonna utilizziamo il comando `cbind()`:

```
> Area<-c(84,544,358)
> Area
[1]  84 544 358
> nazioni.dati<-cbind(nazioni.dati,Area)
> variabili<-c(variabili,"Area")
> dimnames(nazioni.dati)<-list(nazioni,variabili)
> nazioni.dati
          PIL Pop Inflazione Area
 Austria  197   8        1.8   84
 Francia 1355  58        1.7  544
Germania 2075  81        1.8  358
```

Per aggiungere una nuova riga usiamo invece il comando `rbind()`:

```
> Svizzera<-c(265,7,1.8,41)
> Svizzera
[1] 265.0   7.0   1.8  41.0
> nazioni.dati<-rbind(nazioni.dati,Svizzera)
> nazioni<-c(nazioni,"Svizzera")
> dimnames(nazioni.dati)<-list(nazioni,variabili)
> nazioni.dati
          PIL Pop Inflazione Area
 Austria  197   8        1.8   84
 Francia 1355  58        1.7  544
Germania 2075  81        1.8  358
Svizzera  265   7        1.8   41
```

Oltre al comando `matrix()` esiste anche il comando `array()` per generare matrici. Il comando `array()` serve anche per generare tensori, matrici cioè che hanno più di due dimensioni (più di due indici):

```
> x<-array(1:24,c(3,4,2))
> x

, , 1
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2
     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24
> x[,2,]
     [,1] [,2]
[1,]    4   16
[2,]    5   17
[3,]    6   18
```

il tensore x ha tre dimensioni. Un comando molto utile è `apply()`. Esso serve per eseguire operazioni sulle righe o sulle colonne di una matrice. Ad esempio il seguente comando calcola il valore massimo per ogni colonna:

```
> apply(nazioni.dati,2,max)
      PIL        Pop Inflazione     Area
   2075.0       81.0        1.8      544.0
>
```

Capitolo 3

Lettura di dati da *file* esterni

Usando il comando `scan` è possibile leggere il contenuto di un file e insieme al comando matrix costruire una matrice. I dati devono essere scritti con un editor di testo e salvati in modalità ASCII. Ogni numero sarà separato dal successivo da almeno uno spazio. Se il *file*) che contiene i vostri dati si chiama `esempio1.txt` e il suo contenuto è il seguente:

```
 197    8  1.8
1355   58  1.7
2075   81  1.8
```

allora con l'insieme di comandi che segue, nella variabile E abbiamo i nostri dati disponibili:

```
> E<-matrix(scan(file="esempio1.txt"),ncol=3,byrow=TRUE)
> E
     [,1] [,2] [,3]
[1,]  197    8  1.8
[2,] 1355   58  1.7
[3,] 2075   81  1.8
```

È possibile attraverso la funzione `count.fields()` ottenere per ogni colonna il numero di osservazioni disponibili, ad esempio:

```
> count.fields("‘esempio1.txt")
[1] 3 3 3
>
```

Un altro modo per accedere ai dati è con uso della funzione `read.table()` che permette di leggere un intero *data-frame* con un formato speciale:

- La prima riga contiene i nomi delle colonne
- Ogni riga contiene nella prima posizione il nome della

riga e un numero di elementi pari al numero di variabili, ad esempio leggiamo il contenuto del *file* `nazioni.txt`:

```
> read.table("nazioni.txt")
          PIL Pop Inflazione Area     EU
Austria    NA   8        1.8   84     EC
Francia  1355  58        1.7  544     EC
Germania 2075  81        1.8  358     EC
Svizzera  265   7        1.8   41 non-EU
>
```

Se non si vogliono specificare i nomi delle righe si userà l'opzione `header=TRUE`. Se il *file* di dati non è nella stessa directory in cui si stà eseguendo R è necessario specificare il percorso completo. Per gli utenti di linux questo significa utilizzare la consueta notazione, ad esempio: `file="~ /esercitazioni/statistica/nazioni.t` mentre per gli utenti di windows è necessario raddoppiare la barra, cioè: `file="c:`
`esercitazioni`
`statistica`
`nazioni.txt"`.

Esiste infine il comando `data()` che consente di visualizzare tutti i dataset già disponibili in R , è sufficiente usare il nome del dataset come argomento della funzione per ottenere il dataset richiesto.

Capitolo 4

Data-Frame, Liste e Valori Mancanti

I *data-frame* sono un tipo di struttura di dati simile alle matrici. Hanno però la proprietà aggiuntiva di permettere di legare assieme vettori di diverso tipo. Creiamo un vettore di stringhe che dica quali dei paesi che stiamo studiando appartengono all'Unione Europea (Austria, Francia e Germania) e quali no (Svizzera).

```
> EU<-c("EC","EC","EC","non-EU")
>
```

Aggiungiamo una colonna EU alla matrice di dati sulle nazioni.

```
> nazioni.dati.1<-cbind(nazioni.dati,EU)
> nazioni.dati.1
         PIL    Pop  Inflazione Area
Austria  "197"  "8"  "1.8"      "84"  "EC"
Francia  "1355" "58" "1.7"      "544" "EC"
Germania "2075" "81" "1.8"      "358" "EC"
Svizzera "265"  "7"  "1.8"      "41"  "non-EU"
>
```

Visto che nella matrice era presente una colonna di stringhe, i vettori numerici sono stati tutti convertiti in stringhe e non possiamo più fare calcoli che riguardano i vettori numerici. Ad esempio:

```
> apply(nazioni.dati.1,2,max)
```

```
Error in max(..., na.rm = na.rm) : invalid argument type
>
```

L'errore discende dal fatto che la funzione `max` non può essere applicato a stringhe. Se ci servono sia vettori numerici che non numerici dobbiamo usare i *data-frame* e non le matrici.

```
> nazioni.frame<-data.frame(nazioni.dati,EU,row.names=nazioni)
> nazioni.frame
          PIL Pop Inflazione Area     EU
Austria   197   8        1.8   84     EC
Francia  1355  58        1.7  544     EC
Germania 2075  81        1.8  358     EC
Svizzera  265   7        1.8   41 non-EU
>
```

I vettori numerici questa volta sono rimasti numerici. Calcoliamo i massimi, come prima, ma solo per i vettori numerici:

```
> apply(nazioni.frame[,1:4],2,max)
       PIL        Pop Inflazione       Area
    2075.0       81.0        1.8      544.0
>
```

Si noti che il modo di accedere agli elementi di un *data-frame* è analogo al modo di accedere gli elementi di una matrice. Altri modi per accedere agli elementi di un *data-frame* sono:

```
> nazioni.frame$PIL
[1]  197 1355 2075  265
> nazioni.frame[,"PIL"]
[1]  197 1355 2075  265
>
```

Per rendere più semplice l'accesso agli elementi di un *data-frame* si può usare la funzione `attach()`:

```
> attach(nazioni.frame)
>
```

In questo modo tutte le colonne di un *data-frame* diventano accessibili direttamente. Ad esempio:

```
> PIL
[1]  197 1355 2075  265
```

L'operazione contraria è `detach()`:

```
> detach()
>
```

Quindi la variabile `PIL` non è più disponibile direttamente:

```
> PIL
Error: Object "PIL" not found
>
```

Le liste permettono di legare assieme dati che non condividono la stessa struttura, come ad esempio matrici, singole stringhe, ecc. In questo senso le liste sono un tipo di struttura di dati ancora più generale dei *data-frame*.

Per vedere come funzionano le liste, creiamo innanzitutto un vettore di stringhe con i nomi dei paesi europei in cui il tedesco è la lingua principale o una delle lingue ufficiali.

```
> lingua.tedesca<-c("Austria","Belgio","Germania",
+ "Liechtenstein","Lussemburgo","Svizzera")
>
```

Creiamo adesso una lista con primo elemento il *data-frame* su cui stavamo lavorando e secondo elemento la lista di nazioni appena creata:

```
> nazioni.lista<-list(nazioni.frame,lingua.tedesca)
>
```

Vediamo il risultato:

```
> nazioni.lista
[[1]]
          PIL Pop Inflazione Area     EU
Austria   197   8        1.8   84     EC
Francia  1355  58        1.7  544     EC
Germania 2075  81        1.8  358     EC
Svizzera  265   7        1.8   41 non-EU
```

```
[[2]]
[1] "Austria"       "Belgio"        "Germania"      "Liechtenstein"
[5] "Lussemburgo"   "Svizzera"

>
```

Per accedere alle informazioni della lista non è possibile usare il nome del vettore orginale come si può vedere dal seguente esempio:

```
> nazioni.lista$lingua.tedesca
NULL
>
```

Mentre:

```
> nazioni.lista[[1]]["Austria","PIL"]
[1] 197
>
```

Le doppie parentesi quadre vengono usate per accedere ai vari elementi della lista. Proviamo a trovare il minimo degli elementi numerici del *data-frame* che costituisce il primo elemento della lista:

```
> apply(nazioni.lista[[1]][,1:4],2,min)
       PIL       Pop Inflazione       Area
     197.0       7.0        1.7       41.0
>
```

Talvolta accade che non tutte le informazioni richieste sono disponibili. Un'informazione non disponibile viene indicata da R con `NA`, che significa *"Not Available"*. Ad esempio, se il PIL per l'Austria non fosse disponibile:

```
> nazioni.frame$PIL[1]<-NA
>
```

Otteniamo così:

```
> nazioni.frame$PIL
[1]   NA 1355 2075  265
>
```

Naturalmente operazioni su vettori che contengono valori mancanti, daranno come risultato un valore mancante:

```
> min(nazioni.frame$PIL)
[1] NA
>
```

Per applicare un'operazione matematica ai soli valori disponibili usiamo la funzione `is.na` che restituisce `TRUE` se il valore è mancante e `FALSE` altrimenti.

```
> pil.na<-is.na(nazioni.frame$PIL)
> pil.na
[1]  TRUE FALSE FALSE FALSE
>
```

Calcoliamo il minimo dei valori disponibili, eliminando dal vettore `PIL` i valori mancanti con `!pil.na`:

```
> min(nazioni.frame$PIL[!pil.na])
[1] 265
```

Infatti, poichè `pil.na` ha `TRUE` in corrispondenza dei valori disponibili e `FALSE` in corrispondenza dei valori non disponibili, vengono selezionati solo i valori disponibili.

Capitolo 5

Costruzione di grafici

In questa sezione vedremo quali sono le principali rappresentazioni grafiche. Per illustrarle utilizziamo un insieme di dati riguardante misure della temperatura corporea in gradi centigradi e della frequenza cardiaca misurata come numero di battiti al minuto suddivise per genere. Questo insieme di dati è memorizzato nel *file* `"normtempc.txt"`. Usiamo le istruzioni `scan()` e `matrix()` per acquisire le misurazioni.

```
>  normtemp<-matrix(scan(file="normtempc.txt"),
+  ncol=3,byrow=TRUE)
Read 390 items
>
```

Possiamo vedere una parte delle misurazioni con la seguente istruzione che stampa le prime 5 osservazioni.

```
> normtemp[1:5,]
     [,1] [,2] [,3]
[1,] 35.7    1   70
[2,] 35.9    1   71
[3,] 36.1    1   74
[4,] 36.1    1   80
[5,] 36.2    1   73
>
```

Vediamo quante misurazioni sono state eseguite sui maschi e quanti sulle femmine. Per far questo calcoliamo ad esempio la frequenza relativa percentuale delle due modalità della variabile genere.

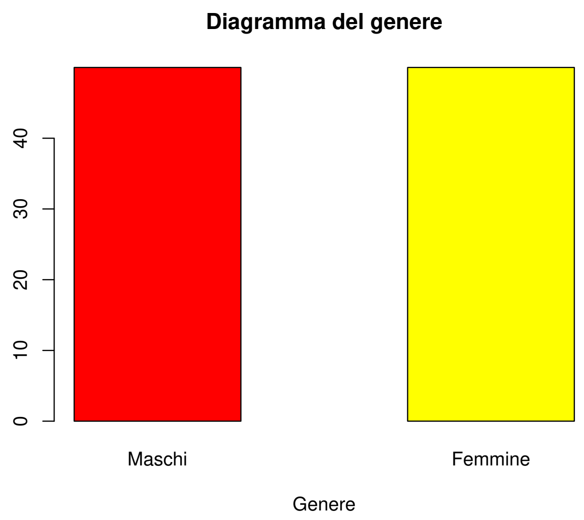

Figura 5.1: Diagramma a rettangoli separati.

```
> n.maschi<-sum(genere==1)/length(genere)*100
> n.femmine<-sum(genere==2)/length(genere)*100
>
```

Rappresentiamo graficamente la variabile genere con l'istruzione `barplot()` che serve per realizzare diagrammi a rettangoli separati, adatto a variabili che sono su scala qualitativa (nominale o mutabile).

```
> barplot(c(n.maschi,n.femmine),space=c(1,0),xlab="Genere",
+ ylab="Frequenza relativa %",main="Diagramma del genere",
+ names.arg=c("Maschi","Femmine"))
>
```

L'opzione `space=` serve per definire la distanza tra i rettangoli. Ovviamente, a differenza dell'istogramma, qui l'area non è importante, ma sono importanti le altezze. Come si può facilmente vedere, vi sono un ugual numero di misurazioni riferite ai maschi e alle femmine. Il grafico è in figura 5.1. È da notare che la variabile genere non è oggetto di studio, infatti chi ha svolto le misurazioni ha fissato il numero delle misurazioni relative ai maschi e alle femmine.

Creiamo tre vettori contenenti le tre diverse variabili.

```
> temperatura<-normtemp[,1]
> genere<-normtemp[,2]
> battiti<-normtemp[,3]
>
```

Una forma molto semplice per rappresentare una variabile quantitativa è il grafico ramo-foglia (*stem-leaf*), ad esempio per la variabile temperatura corporea abbiamo:

```
> stem(temperatura)

  The decimal point is 1 digit(s) to the left of the |

  356 | 0
  358 | 000
  360 | 000
  362 | 000000000000
  364 | 0000000000
  366 | 0000000000000000000000000
  368 | 000000000000000000000000000
  370 | 0000000000000000000000000000
  372 | 0000000000000
  374 | 00000
  376 | 0
  378 | 0
  380 |
  382 | 0

>
```

dove si vede che la classe modale è costituita dalla classe $37.0 \vdash 37.2$. Una rappresentazione grafica alternativa e forse più usata è l'istogramma.

```
> hist(temperatura,main="Istogramma delle temperatura",
+ xlab="Temperatura in gradi centigradi")
>
```

Diversamente dal `barplot()` nell'istruzione `hist()` l'area rappresenta la frequenza della classe. Il grafico è riportato in figura 5.2.

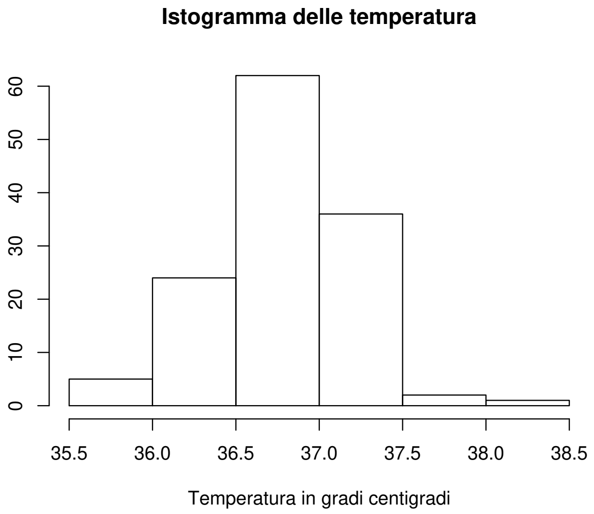

Figura 5.2: Istogramma delle temperature.

R decide automaticamente il numero di classi, in questo caso ha generato 13 classi. Il numero di classi può essere facilmente fissato usando l'opzione `breaks=`:

```
> hist(temperatura,main="Istogramma delle temperatura",
+ breaks=20,
+ xlab="Temperatura in gradi centigradi")
>
```

Come si vede dalla figura 5.3 aumentare il numero di classi porta ad un maggiore dettaglio.

Specificando un vettore di valori nell'opzione `breaks=` definiamo gli estremi delle classi, in questa maniera precisiamo sia l'ampiezza delle classi (che può risultare diversa) sia il numero di classi, inoltre con l'opzione `probability=FALSE` otteniamo un grafico che usa le frequenze assolute invece delle frequenze relative:

```
> hist(temperatura,main="Istogramma delle temperatura",
+ breaks=c(35.5,36.5,37.5,38.5),
+ probability=FALSE,xlab="Temperatura in gradi centigradi")
>
```

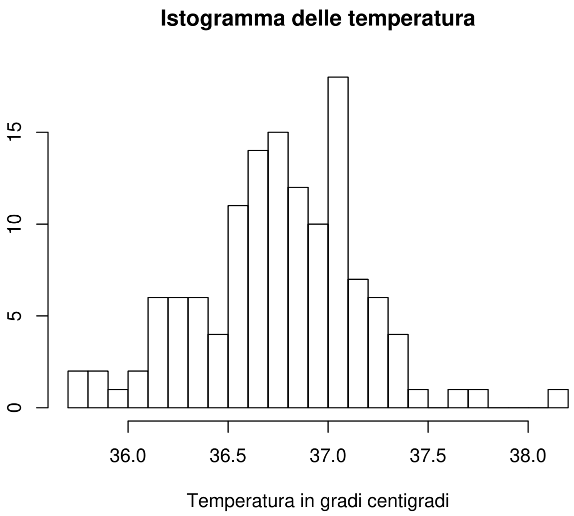

Figura 5.3: Istogramma con 20 classi.

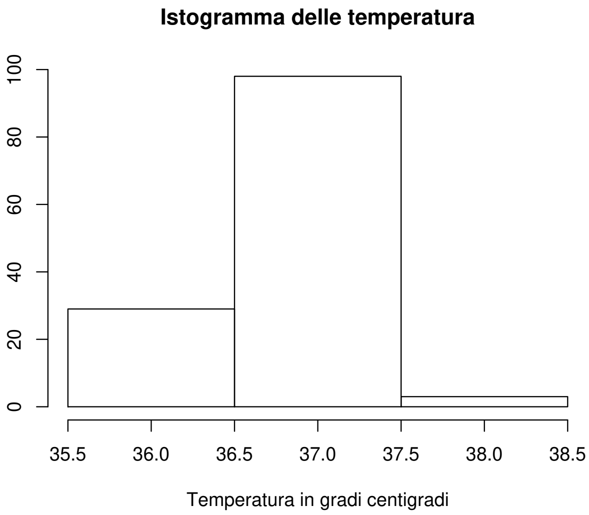

Figura 5.4: Istogramma definito a partire dalle classi e basato sulle frequenza assolute.

Nella figura 5.4 abbiamo l'istogramma definito con tre classi.

Se usiamo l'opzione `plot=FALSE` il grafico non verrà presentato sulla finestra grafica, questa opzione è utile se vogliamo memorizzare in una variabile le informazioni relative ad un istogramma:

```
> dati.hist<-hist(temperatura,breaks=c(35.5,36.5,37.5,38.5),
+ probability=FALSE,plot=FALSE)
> str(dati.hist)
List of 4
 $ breaks   : num [1:4] 35.5 36.5 37.5 38.5
 $ counts   : int [1:3] 25 101 4
 $ rel.freqs: num [1:3] 0.005225 0.021108 0.000836
 $ mids     : num [1:3] 36 37 38
>
```

In `$breaks` ci sono gli estremi delle classi, in `$counts` ci sono le frequenze assolute che cadono all'interno di ogni classe, mentre in `$rel.freqs` ci sono le frequenze relative, anche se a volte questa variabile non sembra calcolata correttamente, quindi si presti particolare attenzione al suo utilizzo. In `$mids` sono riportati i valori centrali delle classi.

Può risultare interessante verificare se esiste una differenza tra la distribuzione delle temperature registrate per i maschi e la distribuziuone per le femmine. Suddividiamo le misurazioni relative alla temperatura in due vettori distinti:

```
> temperatura.maschi<-temperatura[genere==1]
> temperatura.femmine<-temperatura[genere==2]
>
```

Usando i comandi `summary()` e `boxplot.stats()` vediamo se vi sono differenze tra le due distribuzioni. In particolare i valori forniti da `boxplot.stats()` si riferiscono a `$stats`: l'estremo inferiore del baffo, l'estremo inferiore della scatola, il valore centrale, l'estremo superiore della scatola e l'estremo superiore del baffo; `$n`: il numero di osservazioni; `$out`: il valore di tutte le osservazioni che sono al di fuori dei baffi.'

```
> sum.temp.maschi<-summary(temperatura.maschi)
> sum.temp.maschi
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

```
  35.70   36.40   36.70   36.73   37.00   37.50
> boxplot.stats(temperatura.maschi,coef = 1.5)
$stats
[1] 35.7 36.4 36.7 37.0 37.5

$n
[1] 65

$conf
[1] 36.58242 36.81758

$out
real(0)

>

> sum.temp.femmine<-summary(temperatura.femmine)
> sum.temp.femmine
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  35.80   36.70   36.90   36.89   37.10   38.20
> boxplot.stats(temperatura.femmine,coef = 1.5)
$stats
[1] 36.2 36.7 36.9 37.1 37.7

$n
[1] 65

$conf
[1] 36.82161 36.97839

$out
[1] 35.8 35.9 36.0 37.8 38.2

>
```

La temperatura delle femmine sembra leggermente superiore a quella dei maschi, ma per decidere quanto questa differenza sia realmente significativa è necessario utilizzare una tecnica che vedremo più avanti.

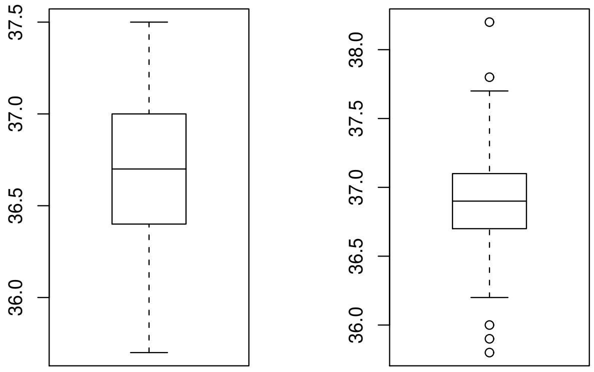

Figura 5.5: Diagrammi a scatola e baffi.

L'istruzione `par()` serve per porre nella finestra grafica più grafici contemporaneamente. Come nell'esempio che segue:

```
> par(mfcol=c(1,2))
>
```

dove rappresentiamo i diagrammi a scatola e baffi separatamente per i maschi e le femmine, il risultato è in figura 5.5.

```
> boxplot(temperatura.maschi)
> boxplot(temperatura.femmine)
>
```

Per confrontare le due distribuzioni è meglio porre i diagrammi su uno stesso asse come segue:

```
> boxplot(temperatura.maschi,temperatura.femmine,
+ names=c("Maschi","Femmine"))
>
```

Il diagramma risultante è in figura 5.6. Oppure per ottenere lo stesso risultato possiamo usare la seguente sintassi che risulta equivalente:

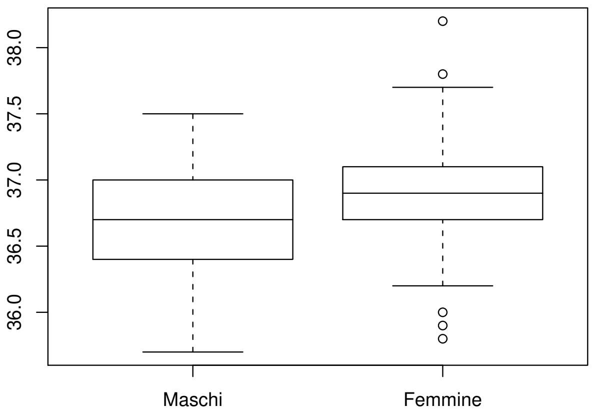

Figura 5.6: Diagramma a scatola e baffi.

```
> boxplot(temperatura ~ genere, names=c("Maschi","Femmine"))
>
```

Dal diagramma a baffi si nota come la distribuzione della temperatura corporea delle femmine tenda ad essere più elevata di quella dei maschi. Differenze si possono trovare anche nella dispersione di queste due distribuzioni.

Infine un'ulteriore maniera per verificare se le distribuzioni della temperatura dei maschi e delle femmine sono oppure no le stesse, usiamo una rappresentazione basata sui quantili:

```
> qqplot(temperatura.maschi,temperatura.femmine,
+ xlab="Maschi",ylab="Femmine")
>
```

Il risultato è riportato in figura 5.7.

Quando siamo in presenza di campioni con numerosità abbastanza diverse (non è il nostro caso) è opportuno usare l'istruzione `nqqplot()` che non è parte di R ma è stata definita esternamente. Per renderla disponibile usiamo l'istruzione `source()`:

```
> source("nqqplot.r")
```

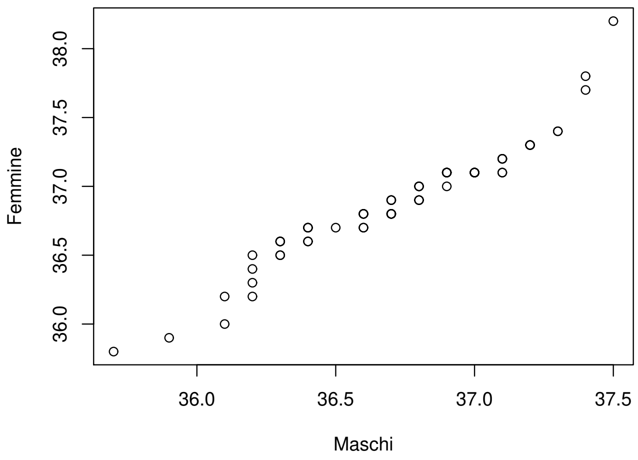

Figura 5.7: Grafico quantile-quantile.

```
> nqqplot(temperatura.maschi,temperatura.femmine,
+ xlab="Maschi",ylab="Femmine")
>
```

Con la funzione `nqqplot()` è possibile tracciare una retta di riferimento, nel caso che le distribuzioni siano uguali, tutti i nti saranno su questa retta. IL risultato è in figura 5.8.

Quando siamo interessati allo studio della relazione tra due variabili quantitative su scala continua, è sempre opportuno rappresentare le osservazioni in un piano cartesiano. In R questo si ottiene facilmente attraverso l'istruzione `plot()`:

```
> plot(battiti,temperatura,ylab="Temperatura del Corpo",
+ xlab="Numero di battiti al minuto")
>
```

ottenendo il grafico in figura 5.9. Come si vede dal grafico non sembra esserci una relazione marcata tra la temperatura corporea e il numero di battiti al minuto.

In questo contesto potrebbe essere interessante vedere dove sono collocate le osservazioni che si riferiscono alle femmine rispetto a quelle dei maschi.

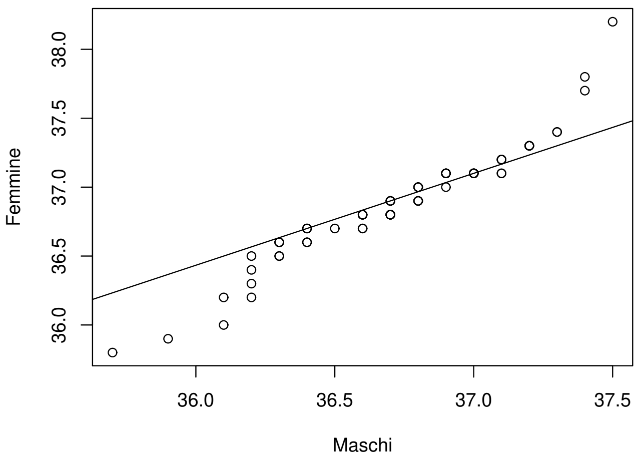

Figura 5.8: QQplot, in ascissa la temperatura corporea dei maschi, in ordinata la temperatura corporea delle femmine.

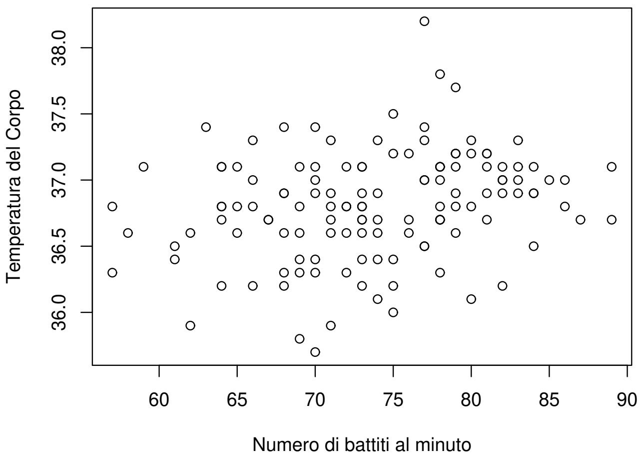

Figura 5.9: Numero di battiti cardiaci, temperatura corporea.

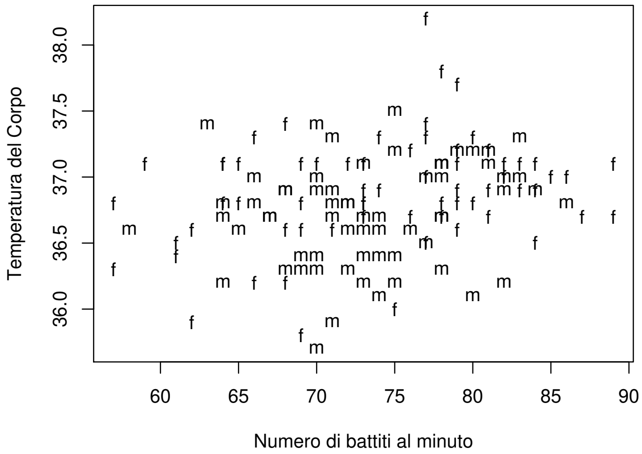

Figura 5.10: Numero di battiti cardiaci, temperatura corporea e sesso.

Per ottenere un grafico di questo genere, riportato in figura 5.10, usiamo il seguente gruppo di istruzioni:

```
> plot(battiti,temperatura,ylab="Temperatura del Corpo",
+ xlab="Numero di battiti al minuto",type="n")
> points(battiti[genere==1],temperatura[genere==1],pch="m")
> points(battiti[genere==2],temperatura[genere==2],pch="f")
>
```

Dal grafico non si evidenzia una marcata differenza tra le osservazioni riferite alle femmine rispetto a quelle riferite ai maschi.

Capitolo 6

Costruzione di tabelle

In questa sezione vediamo alcuni aspetti riguardanti la manipolazione e la costruzione di tabelle di contingenza. Utilizziamo a questo scopo i dati rilevati in una indagine svolta nel 1995 dal Dipartimento di Statistica dell'Università di Padova sulle donne in carriera (donne cioè che svolgono una attività professionale) per conto dell'Associazione Donne e Sviluppo e della Provincia di Padova. Ci interesseremo di tre variabili. Il numero di figli, il grado di istruzione della donna e la sua età. Il grado di istruzione è stato rilevato in 5 diverse categorie:

1 Licenza scuola media o simile

2 Diploma scuola professionale (2 o 3 anni)

3 Diploma scuola secondaria (4 o 5 anni)

4 Laurea

5 Altro

Il nostro interesse è quello di studiare la variabile numero di figli in relazione al grado di istruzione. La terza variabile è l'anno di nascita della donna intervistata.

Leggiamo le tre variabili:

```
> donne<-matrix(scan(file="I:\\statistica\\donne_3.txt"),ncol=3,byrow=TRUE)
Read 708 items
>
```

Diamo un nome alle tre colonne:

```
> dimnames(donne)<-list(NULL,c("figli","istruzione","anno"))
>
```

Le prime osservazioni sono le seguenti:

```
> donne[1:6,]
     figli istruzione anno
[1,]     5          4   46
[2,]     1          3   51
[3,]     1          4   53
[4,]     1          4   50
[5,]     1          4   51
[6,]     1          4   33
>
```

Si noti che nello studio tra le due variabili di interesse l'età della donna intervistata può non essere considerata in prima battuta in quanto le donne intervistate hanno una età media sufficientemente elevata da far pensare che esse abbiano già avuto il numero di figli desiderato:

```
> eta<-95 - donne[,3]
> summary(eta)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   27.0    43.0    49.0    49.7    57.0    82.0
>
```

Come si può vedere anche da un diagramma a scatola e baffi riportato in figura 6.1 ottenuto con la funzione `boxplot(eta)`.

Definiamo due vettori, uno per il numero di figli e uno per il grado di istruzione. Inoltre contiamo quante osservazioni abbiamo:

```
> figli<-donne[,1]
> istruzione<-donne[,2]
> n<-length(figli)
> n
[1] 236
>
```

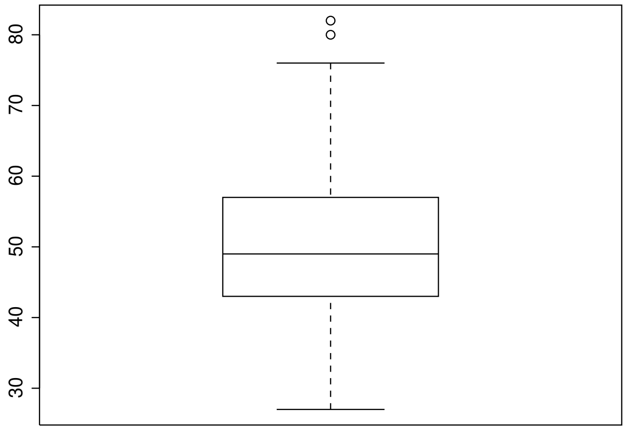

Figura 6.1: Diagramma a scatola e baffi della variabile età.

Per vedere quali sono i valori estremi assunti dalla variabile `figli` usiamo le funzioni `min()` e `max()`:

```
> min(figli)
[1] 0
> max(figli)
[1] 9
>
```

Possiamo calcolare la distribuzione marginale del numero di figli con le istruzioni che seguono:

```
> numero.figli<-c(sum(figli==0),sum(figli==1),sum(figli==2),
+ sum(figli==3),sum(figli==4),sum(figli==5),sum(figli==6),
+ sum(figli==7),sum(figli==8),sum(figli==9))
> numero.figli
 [1] 74 59 69 23  8  2  0  0  0  1
>
```

Alternativamente usiamo la funzione `table()` nel seguente modo:

```
> numero.figli<-table(figli)
```

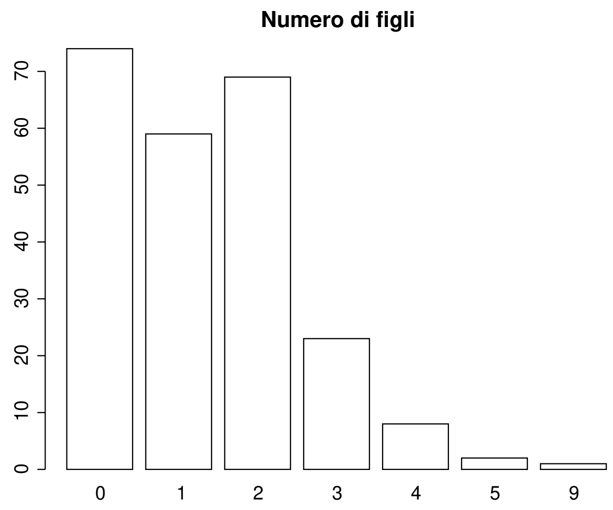

Figura 6.2: Diagramma a rettangoli separati per la variabile numero di figli.

```
> numero.figli
 0  1  2  3  4  5  9
74 59 69 23  8  2  1
>
```

che riporta solamente le modalità che hanno una frequenza assoluta positiva.

Possiamo vedere la distribuzione del numero di figli attraverso la funzione `barplot()`:

```
> barplot(numero.figli,main="Numero di figli",xlab="",
+ ylab="",names.arg=c("0","1","2","3","4","5","9"),col=NULL)
>
```

Nella figura 6.2 è riportato il grafico relativo. Possiamo fare la stessa cosa per ottenere il grafico della variabile grado di istruzione, il grafico è riportato in figura 6.3.

```
> livello.istruzione<-table(istruzione)
> barplot(livello.istruzione,main="Livello di istruzione",
+ xlab="",ylab="",names.arg=c("1","2","3","4","5","9"),
+ col=NULL)
>
```

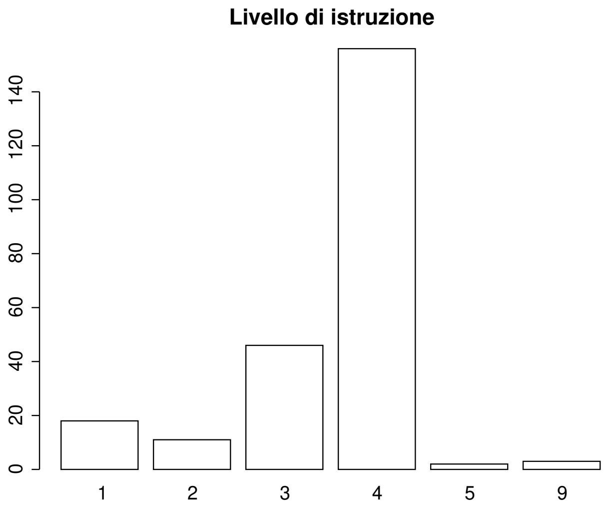

Figura 6.3: Diagramma a rettangoli separati per la variabile grado di istruzione.

Per costruire una tabella di frequenze congiunta è sufficiente riutilizzare la funzione `table()` nel seguente modo:

```
> table(figli,istruzione)
  1 2  3  4 5 9
0 1 2 13 55 0 3
1 2 2 13 42 0 0
2 7 6 15 41 0 0
3 6 1  2 13 1 0
4 2 0  2  3 1 0
5 0 0  1  1 0 0
9 0 0  0  1 0 0
>
```

La modalità 9 si riferisce alla mancata risposta. Per semplificare lo studio possiamo eliminare le osservazioni che si riferiscono alla mancata risposta e alla modalità 5 (altro), non avendo infatti informazioni su quale tipo di istruzione sia posseduto da queste donne diventano osservazioni poco informative e difficilmente trattabili. Infine, queste tre osservazioni non corrisondono a particolari valori della variabile numero di figli e quindi possiamo eliminarle (Per essere precisi si possono controllare questi pochi casi separatamente).

Quindi ridefiniamo i dati di partenza togliendo queste osservazioni:

```
> donne<-donne[(istruzione!=5 & istruzione!=9),]
> figli<-donne[,1]
> istruzione<-donne[,2]
> n<-length(figli)
>
> n
[1] 231
>
```

Otteniamo allora la seguente tabella di contingenza:

```
> tabella<-table(figli,istruzione)
> tabella
  1 2  3  4
0 1 2 13 55
1 2 2 13 42
2 7 6 15 41
3 6 1  2 13
4 2 0  2  3
5 0 0  1  1
9 0 0  0  1
>
```

Definiamo due vettori che contengono le modalità delle due variabili:

```
> mod.figli<-c(0,1,2,3,4,5,9)
> mod.istruzione<-c(1,2,3,4)
>
```

Calcoliamo la distribuzione marginale delle due variabili (si poteva usare la funzione `table()` su ogni singola variabile):

```
> mar.figli<-apply(tabella,1,sum)
> mar.figli
 0  1  2  3  4  5  9
71 59 69 22  7  2  1
```

```
>
> mar.istruzione<-apply(tabella,2,sum)
> mar.istruzione
  1   2   3   4
 18  11  46 156
>
```

La distribuzione delle frequenze relative condizionate della variabile numero di figli dato la varibiale grado di istruzione è semplicemente calcolata utilizzando la funzione `t()` e la divisione elemento per elemento:

```
> figli.dato.istruzione<-t(t(tabella)/mar.istruzione)
> figli.dato.istruzione
           1         2          3           4
0 0.05555556 0.1818182 0.28260870 0.352564103
1 0.11111111 0.1818182 0.28260870 0.269230769
2 0.38888889 0.5454545 0.32608696 0.262820513
3 0.33333333 0.0909091 0.04347826 0.083333333
4 0.11111111 0.0000000 0.04347826 0.019230769
5 0.00000000 0.0000000 0.02173913 0.006410256
9 0.00000000 0.0000000 0.00000000 0.006410256
>
```

Già una ispezione di questa tabella mostra come vi sia una relazione tra il grado di istruzione e il numero di figli. Per rappresentare queste quattro distribuzioni condizionate possiamo usare la funzione `par(mfcol=)` che specifica il numero di grafici da porre in una stessa finestra grafica. Il grafico relativo è presente in figura 6.4.

```
> par(mfcol=c(2,2))
>
> barplot(figli.dato.istruzione[,1]*n,main="Numero di figli",
+ xlab="Condizionato a livello di istruzione 1",ylab="",
+ names.arg=c("0","1","2","3","4","5","9"),col=NULL)
>
> barplot(figli.dato.istruzione[,3]*n,main="Numero di figli",
+ xlab="Condizionato a livello di istruzione 3",ylab="",
+ names.arg=c("0","1","2","3","4","5","9"),col=NULL)
>
```

```
> barplot(figli.dato.istruzione[,2]*n,main="Numero di figli",
+ xlab="Condizionato a livello di istruzione 2",ylab="",
+ names.arg=c("0","1","2","3","4","5","9"),col=NULL)
>
> barplot(figli.dato.istruzione[,4]*n,main="Numero di figli",
+ xlab="Condizionato a livello di istruzione 4",ylab="",
+ names.arg=c("0","1","2","3","4","5","9"),col=NULL)
>
```

Come di vede facilmente la forma delle distribuzioni condizionate cambia notevolmente al mutare del valore della variabile condizionante. Ricordiamo che nel caso di indipendenza stocastica, le distribuzioni condizionate dovrebbero essere uguali al variare del valore della variabile condizionante.

Se consideriamo la variabile numero di figli, una variabile in cui la distanza tra le modalità abbia un senso, e che tale distanza sia la stessa per ogni coppia di modalità contigue allora possiamo calcolare la media e la varianza per riassumere la forma della distribuzione.

```
> media.figli<-as.vector((mod.figli%*%mar.figli))/
+ sum(mar.figli)
> media.figli
[1] 1.341991
> var.figli<-as.vector(((mod.figli-media.figli)^2)%*%mar.figli)/
+ sum(mar.figli)
> var.figli
[1] 1.558367
>
```

dove si usa la fuzione `as.vector()` per ottenere come risultato uno scalare invece che una matrice di dimensione 1×1 (Questo al fine di risolvere un problema di R). Ma più interessante risulta il calcolo delle medie e delle varianza condizionate:

```
> media.figli.dato.istruzione<-mod.figli%*%figli.dato.istruzione
> media.figli.dato.istruzione
            1        2        3        4
[1,] 2.333333 1.545455 1.347826 1.211538
>
```

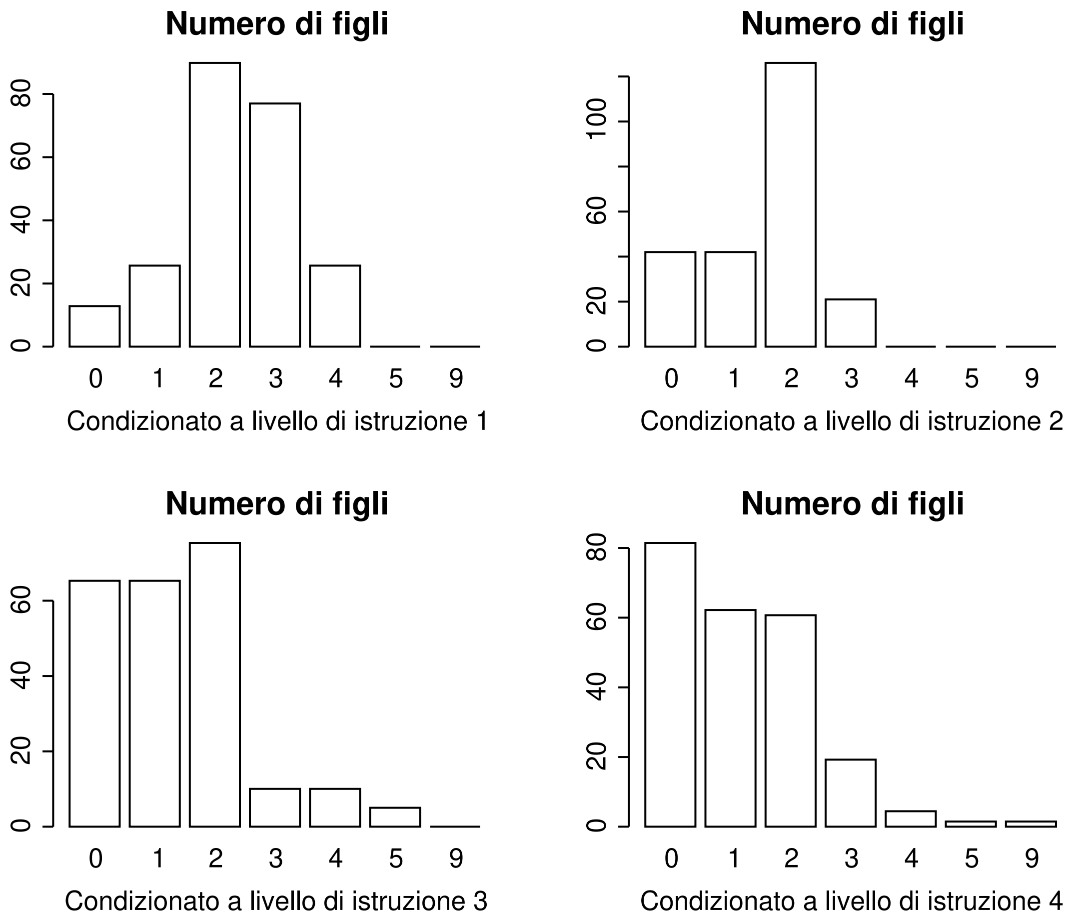

Figura 6.4: Diagramma a rettangoli separati per le distribuzioni condizionate del numero di figli dato il grado di istruzione.

È facile notare che le medie condizionate sono decrescenti rispetto alla variabile grado di istruzione.

Per valutare quanta variabilità della variabile numero di figli sia spiegata dalla variabile grado di istruzione calcoliamo la varianza tra le classi, che è la varianza delle medie condizionate:

```
> var.tra.figli.dato.istruzione<-as.vector((
+ (media.figli.dato.istruzione-media.figli)^2)
+ %*%mar.istruzione)/sum(mar.istruzione)
> var.tra.figli.dato.istruzione
[1] 0.09004934
>
```

Il rapporto tra questa varianza e la varianza della variabile numero di figli è un buon indice della variabilità spiegata:

```
> var.tra.figli.dato.istruzione/var.figli*100
[1] 5.778444
>
```

Il valore molto piccolo indica che la variabilità spiegata è poca e quindi la dipendenza in media è molto debole. A conferma di questa conclusione possiamo costruire un grafico in cui rappresentiamo le medie condizionate insieme ad un intervallo costruito a partire dalle varianze condizionate.

Calcoliamo le varianze condizionate nel seguente modo:

```
> media2.figli.dato.istruzione<-mod.figli^2%*%figli.dato.istruzione
> var.int.figli.dato.istruzione<-media2.figli.dato.istruzione -
+ media.figli.dato.istruzione^2
>
> var.int<-as.vector(var.int.figli.dato.istruzione
+ %*%mar.istruzione)/sum(mar.istruzione)
>
> var.int
[1,] 1.468317
>
```

Con l'ultima riga di istruzione abbiamo calcolato anche la varianza interna alle classi come media pesata delle varianze condizionate. Questa varianza sommata alla varianza tra le classi deve darci il valore della varianza della variabile numero di figli.

Calcoliamo la deviazione standard condizionata:

```
> dev.int.figli.dato.istruzione<-sqrt(var.int.figli.dato.istruzione)
> dev.int.figli.dato.istruzione
     1         2        3        4
[1,] 1 0.8907235 1.183535 1.260899
>
```

e infine i due estremi dell'intervallo:

```
> p1.figli.dato.istruzione<-media.figli.dato.istruzione -
+ dev.int.figli.dato.istruzione
> p2.figli.dato.istruzione<-media.figli.dato.istruzione +
+ dev.int.figli.dato.istruzione
>
```

Rappresentiamo in un grafico le modalità della varibiale grado di istruzione con le medie condizionate del numero di figli:

```
> plot(mod.istruzione,media.figli.dato.istruzione,
+ xlab="Livello di istruzione",ylab="Numero di figli dato
+ il livello di istruzione",ylim=c(0,4))
>
```

poi aggiungiamo attraverso la funzione `lines()` gli intervalli basati sugli estremi `p1.` e `p2.` nel seguente modo:

```
> lines(c(1,1),c(p1.figli.dato.istruzione[1],
+ p2.figli.dato.istruzione[1]))
> lines(c(2,2),c(p1.figli.dato.istruzione[2],
+ p2.figli.dato.istruzione[2]))
> lines(c(3,3),c(p1.figli.dato.istruzione[3],
+ p2.figli.dato.istruzione[3]))
> lines(c(4,4),c(p1.figli.dato.istruzione[4],
+ p2.figli.dato.istruzione[4]))
```

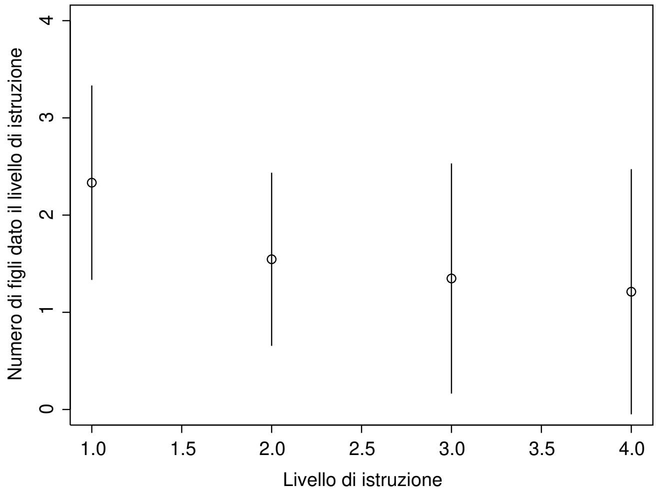

Figura 6.5: Grafico delle medie condizionate e intervalli basati sulle varianze condizionate.

Il risultato è riportato in figura 6.5. Si osservi che ogni media condizionata è all'interno di ogni intervallo. Si ha quindi conferma che non vi è una forte dipendenza in media. Questo fatto non significa però come abbiamo già notato in figura 6.4 che vi sia indipendenza stocastica. Infatti se calcoliamo l'indice Chi-quadrato possiamo valutare la presenza di una dipendenza stocastica.

Calcoliamo innanzitutto, la tabella di contingenza delle frequenze attese, utilizzando le distribuzioni marginali delle due variabili.

```
> tabella.attesa<-matrix(mar.figli,nrow=7)%*%
+ matrix(mar.istruzione,ncol=4)/n
> tabella.attesa
           [,1]       [,2]       [,3]       [,4]
[1,] 5.53246753 3.38095238 14.1385281 47.9480519
[2,] 4.59740260 2.80952381 11.7489177 39.8441558
[3,] 5.37662338 3.28571429 13.7402597 46.5974026
[4,] 1.71428571 1.04761905  4.3809524 14.8571429
[5,] 0.54545455 0.33333333  1.3939394  4.7272727
[6,] 0.15584416 0.09523810  0.3982684  1.3506494
[7,] 0.07792208 0.04761905  0.1991342  0.6753247
```

```
>
```

Infine calcoliamo l'indice:

```
> chi.quadrato<-sum(((tabella-tabella.attesa)^2)/tabella.attesa)
> chi.quadrato
[1] 29.95831
>
```

Da cui si può concludere che vi è dipendenza stocastica tra le due variabili.

Esercizio: Proponiamo un esercizio basato sui dati del questionario sulle donne in carriera. Nel *file:* `"I:\\statistica\\donne_2.txt"` è riportato congiuntamente il grado di istruzione della donna e del marito. Si costruisca una tabella di contingenza e si studi la relazione tra il grado di istruzione del marito e quello della moglie. (Si valuti la necessità di riclassificare opportunamente le variabili).

Esercizio: I dati che seguono riguardano la verifica della continuità del conduttore di protezione contenuti in tre diversi macchinari. Queste misure servono per verificare la sicurezza elettrica dell'equipaggiamento di centraline elettroniche in conformità alle norme DIN VDE 0113 / EN 60204-1 / IEC 204-1.

Le misure vengono eseguite applicando una corrente continua di 10 Ampere e misurando la caduta di tensione (in mVolt) e la resistenza del conduttore (in $m\Omega$). Il valore limite fissato dalla norma è di 1.9 Volt.

I dati sono contenuti nel *file*: `"I:\\statistica\\protezione.txt"`.

1. È possibile usare un modello normale per rappresentare globalmente (senza distinzioni tra i diversi macchinari) il fenomeno.

2. Si calcolino le medie e le varianze condizionate.

3. Si dica quale tipo di macchinario ha il migliore conduttore di protezione.

Mod.	Caduta di tensione(mV)	Resist. cond.prot. ($m\Omega$)
1	1200	124,0
1	1600	164,0
1	1300	134,0
1	1500	156,0
1	1400	145,0
1	1400	142,0
1	1300	139,0
1	1300	136,0
1	1400	145,0
2	410	41,30
2	400	40,90
2	900	97,00
2	440	44,30
2	560	56,10
2	570	57,80
2	540	54,20
2	580	58,40
3	470	47,50
3	540	54,50
3	550	55,20
3	570	57,10
3	560	56,30
3	900	99,00
3	680	68,90
3	640	64,40

Capitolo 7

Regressione

In questa sezione il modello di regressione viene illustrato attraverso l'uso di alcuni insiemi di dati.

Il primo insieme di dati che utilizziamo riguarda le caratteristiche di alcuni tipi di sigarette. Carichiamo i dati contenuti nel *file* `"I:\\statistica\\sigar.txt"`, costituito da cinque colonne che rappresentano rispettivamente la marca, il condensato (contenuto di catrame in mg), il peso (gr) ed il contenuto di monossido di carbonio (mg) per 25 diverse marche di sigarette.

```
> sigar<-read.table("I:\\statistica\\sigar.txt")
>
> sigar
                V1   V2   V3     V4   V5
1           Alpine 14.1 0.86 0.9853 13.6
2     Benson&Hedges 16.0 1.06 1.0938 16.6
3        BullDurham 29.8 2.03 1.1650 23.5
4       CamelLights  8.0 0.67 0.9280 10.2
5          Carlton  4.1 0.40 0.9462  5.4
6      Chesterfield 15.0 1.04 0.8885 15.0
7      GoldenLights  8.8 0.76 1.0267  9.0
8             Kent 12.4 0.95 0.9225 12.3
9             Kool 16.6 1.12 0.9372 16.3
10             L&M 14.9 1.02 0.8858 15.4
11      LarkLights 13.7 1.01 0.9643 13.0
12        Marlboro 15.1 0.90 0.9316 14.4
13           Merit  7.8 0.57 0.9705 10.0
```

```
14      MultiFilter 11.4 0.78 1.1240 10.2
15    NewportLights  9.0 0.74 0.8517  9.5
16              Now  1.0 0.13 0.7851  1.5
17          OldGold 17.0 1.26 0.9186 18.5
18    PallMallLight 12.8 1.08 1.0395 12.6
19          Raleigh 15.8 0.96 0.9573 17.5
20       SalemUltra  4.5 0.42 0.9106  4.9
21         Tareyton 14.5 1.01 1.0070 15.9
22             True  7.3 0.61 0.9806  8.5
23 ViceroyRichLight  8.6 0.69 0.9693 10.6
24    VirginiaSlims 15.2 1.02 0.9496 13.9
25    WinstonLights 12.0 0.82 1.1184 14.9
>
```

Vogliamo verificare se esiste una relazione tra il monossido di carbonio (variabile dipendente) ed il condensato (variabile esplicativa). Assegniamo i dati sul condensato e quelli sul monossido di carbonio a due vettori distinti.

```
> condens<-sigar[,2]
> monoss<-sigar[,5]
>
```

Per dare un primo sguardo alla relazione tra le due variabili, costruiamo un grafico che rappresenti la nuvola di punti osservati.

```
> plot(condens,monoss,xlab="Condensato",ylab="Monossido",
+ main="Grafico Monossido Condensato")
>
```

Il grafico è in figura 7.1. Come si vede, tra le variabili c'è una relazione lineare abbastanza forte. Risulta inoltre evidente che un'osservazione è al di fuori della nuvola in cui stanno la maggior parte dei punti.

Per visualizzare meglio il tipo di relazione esistente tra le due variabili, possiamo utilizzare la funzione `lowess` che "liscia" la nuvola di punti. Il parametro `f` indica la proporzione di dati che vengono utilizzati di volta in volta per ottenere il valore lisciato. Quindi valori più grandi di `f` producono un maggiore effeto di lisciamento.

```
> smooth<-lowess(condens,monoss,f=1/4)
> lines(smooth)
```

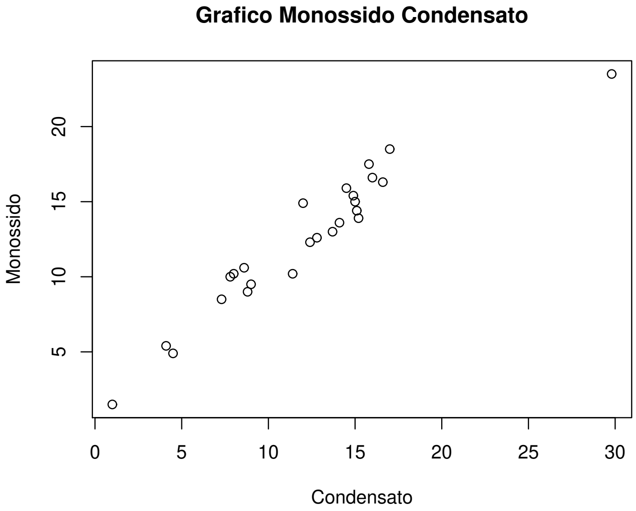

Figura 7.1: Grafico del Monossido rispetto al condensato.

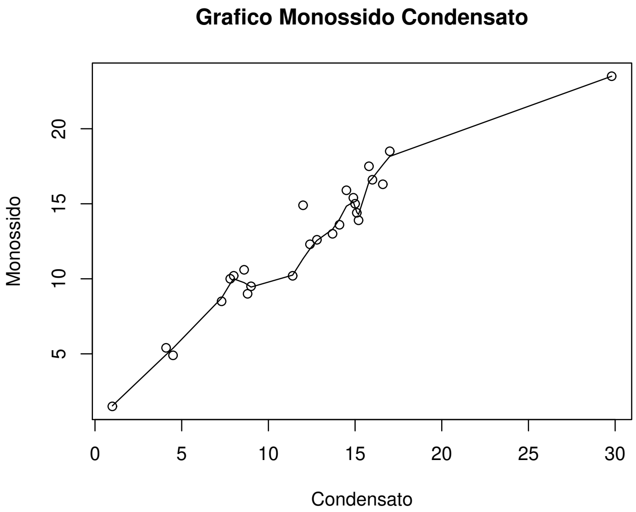

Figura 7.2: Grafico del monossido rispetto al condensato, con nuvola di punti "lisciata" attraverso la funzione `lowess`.

Il risultato è presentato in figura 7.2.

Da questa figura ci viene confermato che, a parte il valore al di fuori del gruppo di osservazioni, una retta dovrebbe rappresentare bene la relazione tra le due variabili.

Calcoliamo i parametri della retta di regressione:

$$monossido = a + b\ condensato$$

È noto che, se chiamiamo x la variabile esplicativa ed y la variabile dipendente, i parametri sono dati da

$$\hat{b} = \frac{\mathrm{Cov}(x, y)}{\mathrm{Var}(x)} \qquad \text{e} \qquad \hat{a} = \mathrm{M}(y) - b\ M(x) \tag{7.1}$$

Calcoliamo innanzitutto le medie delle due variabili:

```
> media.condens<-mean(condens)
> media.condens
[1] 12.216
>
> media.monoss<-mean(monoss)
> media.monoss
[1] 12.528
>
```

Calcoliamo ora le varianze:

```
> n<-length(condens)
>
> var.condens<-sum(condens^2)/n - media.condens^2
> var.condens
[1] 30.81734
>
> var.monoss<-sum(monoss^2)/n - media.monoss^2
> var.monoss
[1] 21.56602
>
```

L'ultima cosa che rimane da calcolare è la covarianza:

```
> covar<-sum((condens-media.condens)*(monoss-media.monoss))/n
> covar
[1] 24.68395
>
```

Facendo adesso riferimento alla 7.1, troviamo:

```
> b<-covar/var.condens
> b
[1] 0.800976
>
> a<-media.monoss - b * media.condens
> a
[1] 2.743278
>
```

La funzione `abline(a,b)` aggiunge al grafico corrente una retta con intercetta `a` e coefficiente angolare `b`.

```
> abline(a,b)
>
```

Il risultato è mostrato in figura 7.3. Una procedura alternativa a quella vista, e più generale, consiste nel costruire la matrice X avente come prima colonna (quella riferita all'intercetta) una colonna di 1 e sulle colonne successive i valori delle variabili esplicative. Si usa poi la formula:

$$\begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix} = \left(X^T X\right)^{-1} X^T \underline{y}$$

In questo caso la seconda colonna della matrice X è costituita dai valori del condensato e $\underline{y}$ è un vettore con i valori del monossido. La procedura è la seguente:

```
> y<-monoss
> X<-cbind(rep(1,n),condens)
>
> param<-solve(t(X) %*% X) %*% t(X) %*% y
>
> param
```

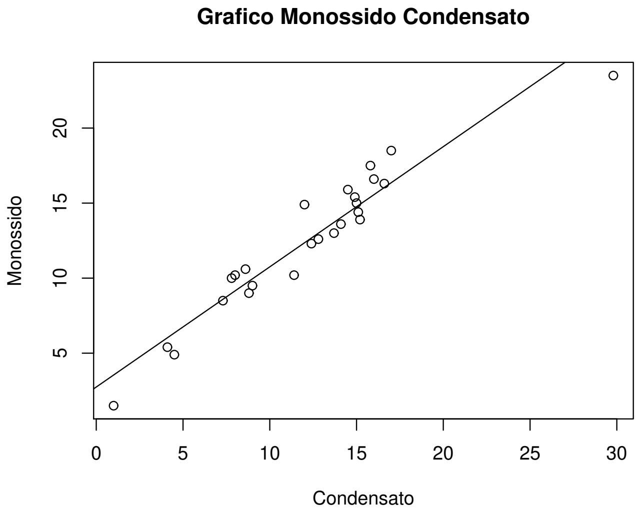

Figura 7.3: Retta del monossido sul condensato.

```
         [,1]
[1,] 2.743278
[2,] 0.800976
>
```

I valori nel vettore `param` coincidono con i valori dei parametri che avevamo già trovato.

Calcoliamo adesso i residui della regressione, cioè la differenza tra i valori osservati ed i valori previsti dalla retta.

```
> prev.monoss<-a + b * condens
> residui<-monoss - prev.monoss
>
> residui[1:5]
[1] -0.4370387  1.0411069 -3.1123615  1.0489147 -0.6272790
>
```

Come si vede, i residui assumono valori attorno allo zero. Il residuo corrispondente alla terza osservazione è, in valore assoluto, molto elevato. Questo residuo è riferito al punto che avevamo già visto essere al di fuori della nuvola. Tracciamo ora il grafico dei residui contro la variabile esplicativa.

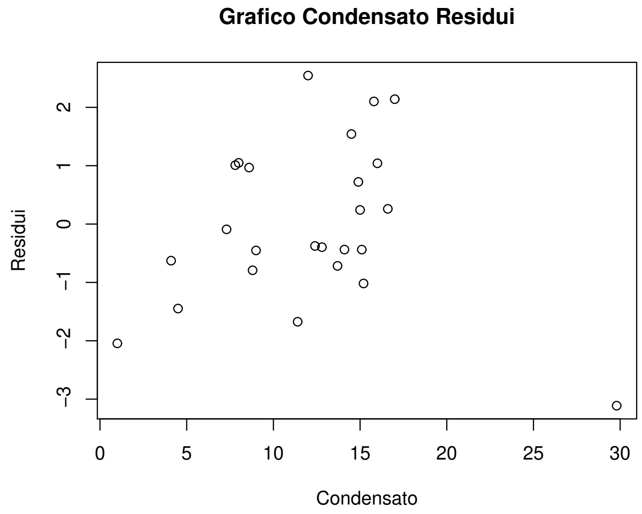

Figura 7.4: Residui della regressione lineare contro i valori della variabile esplicativa.

```
> plot(condens,residui,xlab="Condensato",ylab="Residui",
+ main="Grafico Condensato Residui")
```

Il grafico è riportato in figura 7.4. L'influenza della terza osservazione sulla funzione di regressione può essere messa in evidenza osservando i valori sulla diagonale della matrice di proiezione. Calcoliamo innanzitutto tale matrice:

```
H<-X %*% solve(t(X) %*% X) %*% t(X)
```

Possiamo ottenere i valori sulla diagonale di H con:

```
> diag(H)[1:5]
[1] 0.04460709 0.05858519 0.44132862 0.06307098 0.12549660
>
```

Con l'istruzione:

```
> diag(H)==max(diag(H))
 [1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[25] FALSE
>
```

mettiamo in evidenza che la terza osservazione è quella che esercita maggiore azione di leva sulla funzione di regressione.

◇

Nell'esempio che segue illustriamo un caso di regressione intrinsecamente lineare. Con regressione intrinsecamente lineare ci si riferisce al caso in cui la variabile dipendente y non è legata alla variabile esplicativa x in maniera lineare, ma si può scrivere

$$g(y) = a + bh(x)$$

per opportune funzioni di trasformazione $g(\cdot)$ e $h(\cdot)$.

L'insieme di dati che ci accingiamo a studiare la variabile dipendente è l'intensità della corrente elettrica (misurata in Ampère) prodotta da un mulino a vento e la variabile esplicativa è la velocità del vento. I dati sono nel *file* `"I:\\statistica\\vento.txt"`.

```
> vento<-read.table("I:\\statistica\\vento.txt")
>
> vento
      V1    V2
1   5.00 1.582
2   6.00 1.822
3   3.40 1.057
4   2.70 0.500
5  10.00 2.236
6   9.70 2.386
7   9.55 2.294
8   3.05 0.558
9   8.15 2.166
10  6.20 1.866
11  2.90 0.653
12  6.35 1.930
13  4.60 1.562
14  5.80 1.737
15  7.40 2.088
16  3.60 1.137
17  7.85 2.179
```

```
18  8.80 2.112
19  7.00 1.800
20  5.45 1.501
21  9.10 2.303
22 10.20 2.310
23  4.10 1.194
24  3.95 1.144
25  2.45 0.123
>
```

Salviamo su due vettori distinti la velocità e la corrente prodotta.

```
> vel<-vento[,1]
> dc<-vento[,2]
>
```

Per renderci conto del tipo di relazione tra le due variabili, costruiamo un grafico dell'intensità della corrente prodotta contro la velocità del vento.

```
> plot(vel,dc,xlab="Velocita' del vento",
+ ylab="Intensita' della corrente",
+ main="Grafico Velocita' Intensita'")
>
```

Il grafico è riportato nella figura 7.5.

La figura mostra che la relazione tra le due variabili non è lineare. Un'altra cosa da notare è che, al crescere della velocità del vento, l'intensità della corrente prodotta sembra raggiungere un asintoto.

Vediamo cosa succede se interpoliamo i dati con una retta di regressione nella forma:

$$\text{corrente} = a + b \text{ velocià} \qquad (7.2)$$

Calcoliamo le medie delle due variabili:

```
>  media.vel<-mean(vel)
> media.vel
[1] 6.132
>
> media.dc<-mean(dc)
> media.dc
[1] 1.6096
>
```

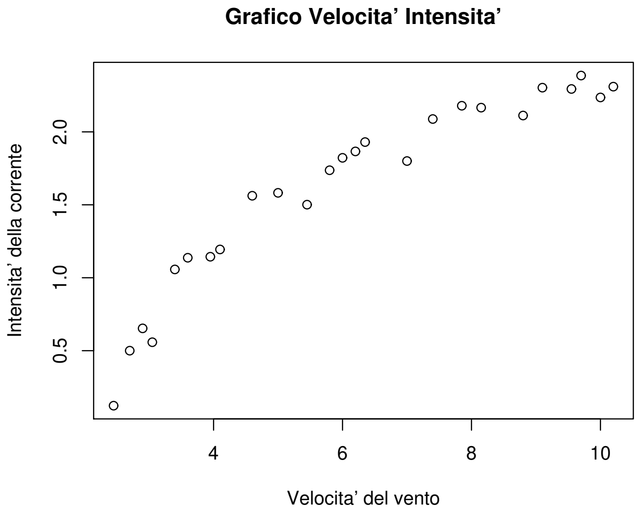

Figura 7.5: Grafico dell'Intensità della corrente prodotta rispetto alla velocità del vento.

Le varianze si trovano invece nel modo che segue:

```
> n<-length(vel)
> n
[1] 25
>
> var.vel<-var(vel)*(n-1)/n
> var.vel
[1] 6.142176
>
> var.dc<-var(dc)*(n-1)/n
> var.dc
[1] 0.4084475
>
```

La covarianza tra la variabile dipendente e la variabile esplicativa è:

```
> covar<-sum((vel-media.vel)*(dc-media.dc))/n
> covar
[1] 1.481179
>
```

Siamo ora in grado di calcolare i due parametri della retta di regressione 7.2.

```
> b<-covar/var.vel
> b
[1] 0.2411489
>
> a<-media.dc - b * media.vel
> a
[1] 0.1308751
>
```

Calcoliamo ora il valore di R^2 per la retta di regressione. Abbiamo bisogno della varianza spiegata e quindi dei valori previsti dalla retta di regressione.

```
> prev.dc<-a + b * vel
>
```

La varianza spiegata è la varianza dei valori previsti e, essendo la media dei valori previsti pari alla media della varianza dipendente, abbiamo:

```
> var.spiegata<-sum((prev.dc - media.dc)^2)/n
> var.spiegata
[1] 0.3571846
>
```

L'indice di determinazione R^2 è il rapporto tra la varianza spiegata e la varianza totale, dove quest'ultima è la varianza della variabile dipendente y:

```
> R2<-var.spiegata/var.dc
> R2
[1] 0.8744932
>
```

Si noti che nel caso di regressione lineare semplice l'indice di determinazione R^2 coincide con il coefficiente di correlazione al quadrato:

```
> r<-covar/sqrt(var.dc * var.vel)
> r^2
[1] 0.8744932
>
```

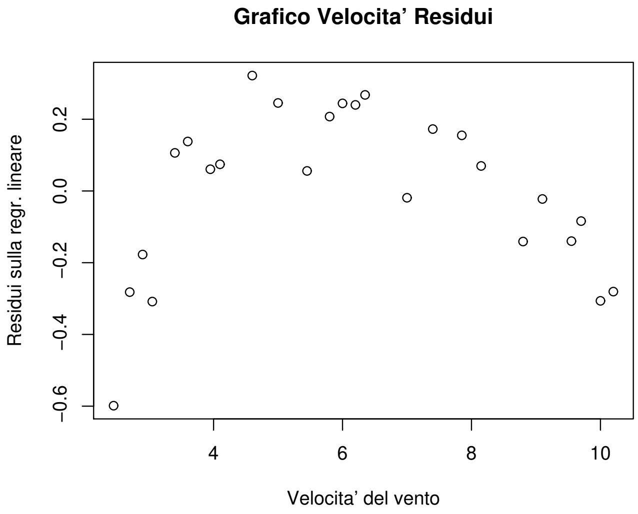

Figura 7.6: Grafico dei residui per il modello lineare rispetto alla velocità del vento.

La retta di regressione spiega l'87% della varianza totale. Calcoliamo ora i residui e ne tracciamo un grafico contro la variabile esplicativa.

```
> res<-dc - prev.dc
>
> plot(vel,res,xlab="Velocita' del vento",
+ ylab="Residui sulla regr. lineare",
+ main="Grafico Velocita' Residui")
>
```

Il grafico è mostrato nella figura 7.6. I residui hanno un andamento niente affatto "casuale" e mettono in evidenza che la relazione tra l'intensità della corrente prodotta e la velocità del vento non è lineare. Questo ci spinge a utilizzare un modello quadratico:

$$\text{corrente} = a + b \text{ velocità} + c \text{ (velocità)}^2 \tag{7.3}$$

Per calcolare i parametri di questo modello costruiamo una matrice X con 1 sulla prima colonna, i valori della velocità del vento sulla seconda colonna e sulla terza colonna gli stessi valori elevati al quadrato.

```
> X<-cbind(rep(1,n),vel,vel^2)
> X[1:5,]
     [,1] [,2]   [,3]
[1,]    1  5.0  25.00
[2,]    1  6.0  36.00
[3,]    1  3.4  11.56
[4,]    1  2.7   7.29
[5,]    1 10.0 100.00
>
```

I parametri del modello 7.3 ottenuti con il metodo dei minimi quadrati sono dati da:

$$\begin{bmatrix} \hat{a} \\ \hat{b} \\ \hat{c} \end{bmatrix} = \left(X^T X\right)^{-1} X^T \underline{y}$$

dove con $\underline{y}$ abbiamo indicato il vettore di valori della variabile dipendente (l'intensità della corrente prodotta). In R :

```
> param<-solve(t(X) %*% X) %*% t(X) %*% dc
> param
            [,1]
[1,] -1.15589824
[2,]  0.72293590
[3,] -0.03812088
>
```

Costruiamo un grafico che mostri come il modello 7.3 interpola i dati e come prevede la corrente generata per valori della velocità del vento maggiori di quelli osservati.

```
> vettx<-seq(2,16,by=0.1)
>
> plot(vel,dc,xlab="Velocita' del vento",
+ ylab="Intensita' della corrente",
+ main="Grafico Velocita' Intensita'")
>
>lines(vettx,param[1]+param[2]*vettx+param[3]*vettx^2)
>
```

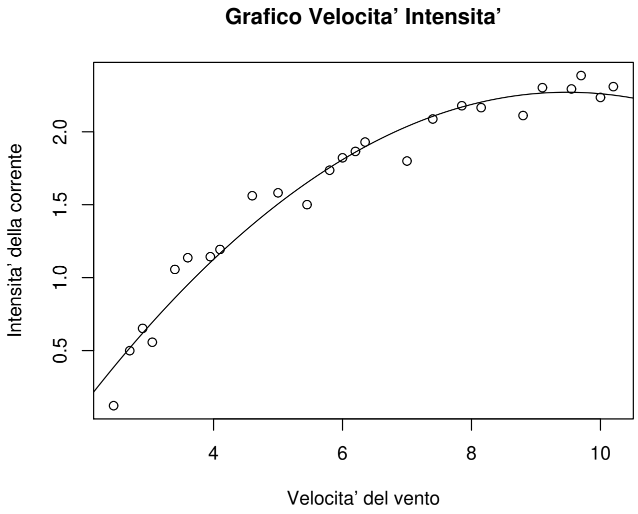

Figura 7.7: Grafico dei valori previsti (linea continua) dal modello di regressione quadratico.

Il grafico è riportato nella figura 7.7. La figura mostra che il modello quadratico interpola bene i dati osservati, ma non tiene conto della tendenza dell'intensità della corrente generata a raggiungere un asintoto. Il modello, prevede che al crescere della velocità l'intensità della corrente generata decresca drasticamente. Questo ci porta a rifiutare anche questo modello.

Un modello che porta al raggiungimento di un asintoto è il seguente:

$$\text{corrente} = a + b\,\frac{1}{\text{velocità}} \tag{7.4}$$

che è lineare nel reciproco della velocità. Calcoliamo il reciproco della velocità e tracciamo un grafico che ne rappresenti la relazione con l'intensità della corrente prodotta.

```
> recvel<-1/vel
>
> plot(recvel,dc,xlab="Reciproco della velocita' del vento",
+ ylab="Intensita' della corrente",
+ main="Grafico Reciproco velocita' Intensita'")
>
```

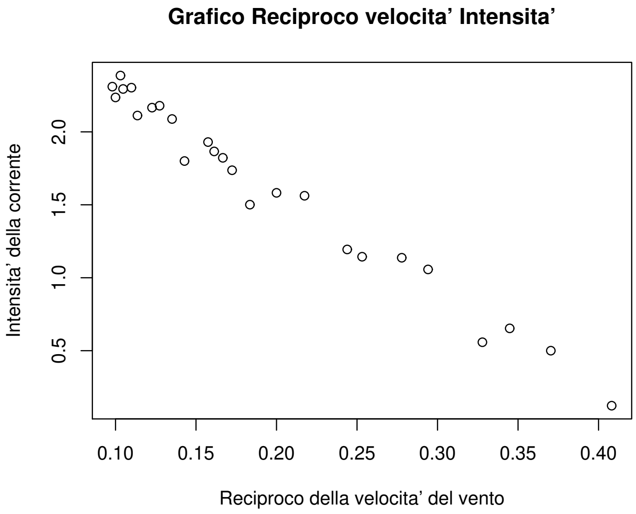

Figura 7.8: Grafico dell'intensità della corrente prodotta rispetto al reciproco della velocità del vento.

Il grafico è mostrato in figura 7.8, dal quale si vede come la relazione tra l'intensità della corrente prodotta e il reciproco della velocità del vento sia lineare. Calcoliamo con il solito procedimento i parametri del modello 7.3.

```
> media.recvel<-mean(recvel)
> media.recvel
[1] 0.1974549
>
> var.recvel<-var(recvel)*(n-1)/n
> var.recvel
[1] 0.008324097
>
> covar<-sum((recvel-media.recvel)*(dc-media.dc))/n
> covar
[1] -0.05772384
>
> b<-covar/var.recvel
> b
[1] -6.934547
```

```
> a<-media.dc - b * media.recvel
> a
[1] 2.97886
>
```

Calcoliamo R^2 per il nuovo modello ricordando che per la regressione lineare semplice $R^2 = \rho^2(dc, recvel)$:

```
> r2<-covar^2/(var.dc*var.recvel)
> r2
[1] 0.9800249
>
```

Il valore risultante è molto elevato e quindi l'adattamento del modello è buono.

Tracciamo infine un grafico dei residui per il modello 7.3 rispetto alla variabile esplicativa di quel modello, cioè il reciproco della velocità del vento.

```
> res<-dc - a + b * recvel
>
> plot(recvel,dc,xlab="Reciproco della velocita' del vento",
+ ylab="Residui",main="Grafico Reciproco velocita' Residui")
>
```

Il grafico dei resiudi è presente in figura 7.9. Questa volta i residui non sembrano avere alcun particolare comportamento e siamo dunque soddisfatti.

Con le istruzioni che seguono viene costruito il grafico in figura 7.10, che mostra l'intensità della corrente elettrica prodotta prevista per i valori della velocità del vento più grandi di quelli osservati:

```
> vettx<-seq(2,16,by=0.1)
>
> plot(vel,dc,xlab="Velocita' del vento",
+ ylab="Intensita' della corrente",
+ main="Grafico Reciproco velocita' Intensita'",
+ xlim=c(0,16),ylim=c(0,3))
>
> lines(vettx,a+b*1/vettx)
>
```

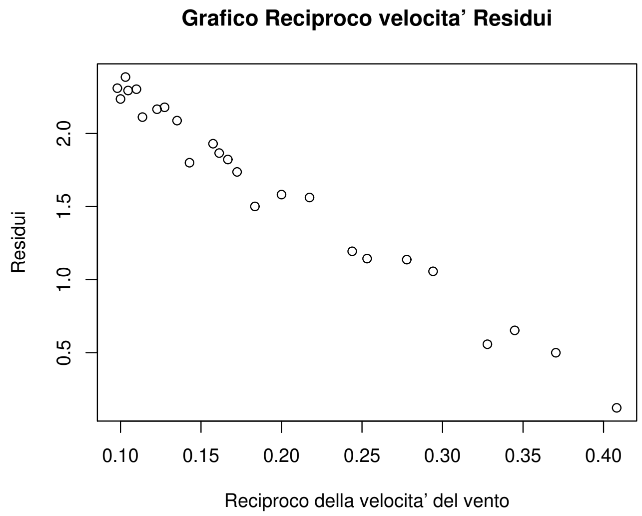

Figura 7.9: Grafico dei residui per il modello lineare nel reciproco della velocità del vento.

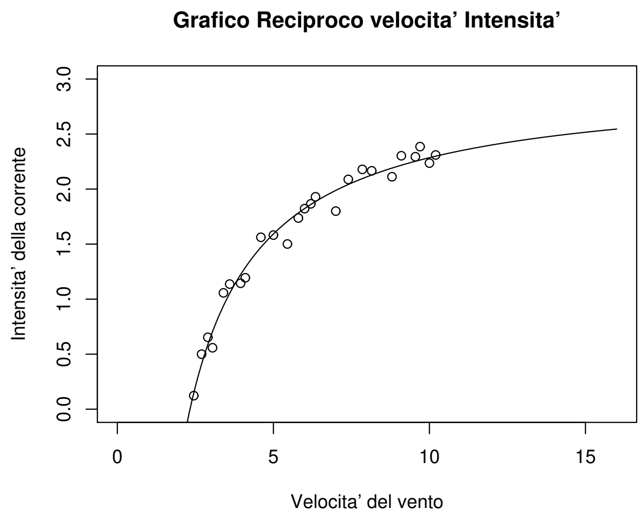

Figura 7.10: Valori previsti dal modello 7.3 per grandi velocità del vento.

In questo caso è evidente che il modello basato sul reciproco della velocità rispetta la tendenza dell'intensità della corrente elettrica generata a raggiungere un asintoto, proprio come volevamo accadesse.

◇

Con questo insieme di dati studiamo la relazione esistente tra il peso corporeo medio di un animale adulto e la sua massa cerebrale. Si vuole capire se per governare un peso corporeo elevato sia necessaria una maggiore massa cerebrale. L'insieme di dati riguarda, il nome dell'animale, il peso corporeo in chilogrammi e la massa cerebrale in grammi. Con il comando `read.table` leggiamo il *file* di dati `"I:\\statistica\\animali.txt"`:

```
> animali<-read.table("I:\\statistica\\animali.txt")
> animali
                    V1       V2     V3
1  Castoro_di_montagna    1.350    8.1
2                Mucca  465.000  423.0
3          Lupo_grigio   36.300  119.5
4                Capra   27.660  115.0
5      Maiale_di_guinea    1.040    5.5
6     Elefante_asiatico 2547.000 4603.0
7                Asino  187.000  419.0
8              Cavallo  521.000  655.0
9      Scimmia_africana   10.000  115.0
10               Gatto    3.300   25.6
11             Giraffe  529.000  680.0
12             Gorilla  207.000  406.0
13                Uomo   62.000 1320.0
14   Elefante_africano 6654.000 5712.0
15      Scimmia_indiana    6.800  179.0
16             Canguro   35.000   56.0
17             Criceto    0.120    1.0
18                Topo    0.023    0.4
19            Coniglio    2.500   12.1
20              Pecora   55.500  175.0
21            Giaguaro  100.000  157.0
22           Scimpanze   52.160  440.0
23               Ratto    0.280    1.9
```

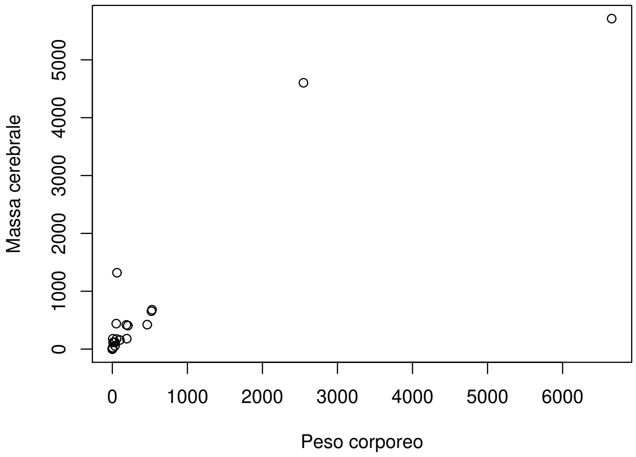

Figura 7.11: Grafico del peso corporeo e della massa cerebrale.

```
24               Talpa    0.122    3.0
25              Maiale  192.000  180.0
```

Costruiamo due vettori, il primo contenente il peso corporeo e il secondo la massa cerebrale:

```
> peso<-animali[,2]
> massa<-animali[,3]
>
```

Osserviamo nel grafico che segue quale possibile modello potrebbe descrivere la relazione:

```
> plot(peso,massa,xlab="Peso corporeo",ylab="Massa cerebrale")
>
```

Il grafico è riportato in figura 7.11. Si vede che una regressione lineare del tipo:

$$massa = \alpha + \beta peso$$

potrebbe essere adeguata. Passiamo allora al calcolo di α e β attraverso il metodo dei minimi quadrati. Costruiamo un vettore di 1 di lunghezza opportuna:

```
> uno<-rep(1,length(peso))
> uno
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
>
```

e la matrice X delle variabili esplicative:

```
> X<-cbind(uno,peso)
>
```

Utilizzando la solita formula otteniamo il valore dei parametri:

```
> parametri<-solve(t(X)%*%X)%*%t(X)%*%massa
> parametri
           [,1]
[1,] 191.227882
[2,]   0.943165
>
```

Aggiungiamo al primo grafico la retta di regressione con il comando:

```
> plot(peso,massa,xlab="Peso corporeo",ylab="Massa cerebrale",
+ xlim=c(0,600),ylim=c(0,1500))
>
> abline(parametri[1,],parametri[2,])
>
```

L'angolo in basso a destra di questo grafico è riportato in figura 7.12. Come si può notare il valore di β è positivo e indica che un corpo più pesante presenta una massa cerebrale più elevata. L'intercetta così elevata tuttavia mostra come i piccoli animali siano mal rappresentati, perché, pesi corporei al di sotto del kilogrammo secondo il modello hanno una massa cerebrale intorno ai 200 grammi. Si veda ad esempio il maiale della guina che pesa circa un kilogrammo ed ha una massa cerebrale pari 5.5 grammi, secondo il modello ha una massa cerebrale pari a 192 grammi.

Calcoliamo, la matrice di proiezione per valutare la presenza di punti leva:

```
> hat<-X%*%solve(t(X)%*%X)%*%t(X)
> dhat<-diag(hat)
> dhat
 [1] 0.04471075 0.04000018 0.04403133 0.04419437 0.04471701 0.13357648
```

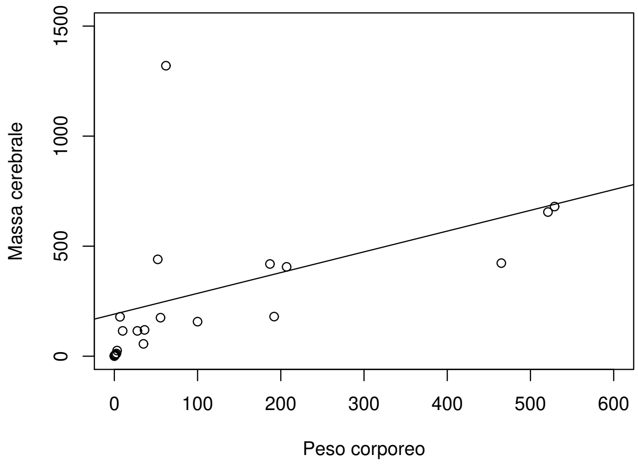

Figura 7.12: Grafico del modello insieme ai dati.

```
 [7] 0.04170738 0.04006116 0.04453767 0.04467145 0.04008095 0.04147286
[13] 0.04356547 0.86839006 0.04460132 0.04405565 0.04473562 0.04473758
[19] 0.04468755 0.04368059 0.04292904 0.04374046 0.04473238 0.04473558
[25] 0.04164713
>
```

Ci serviamo del grafico `dotplot` per osservare chiaramente la presenza di due punti leva, il primo riferito all'elefante asiatico e il secondo riferito all'elefante africano che hanno masse corporee notevolmente superiori ai rimanenti animali.

```
> dotplot(dhat)
>
```

Il grafico è riportato in figura 7.13. Calcoliamo infine le masse cerebrali previste:

```
> massa.prevista<-X%*%parametri
> massa.prevista
          [,1]
 [1,]  192.5012
```

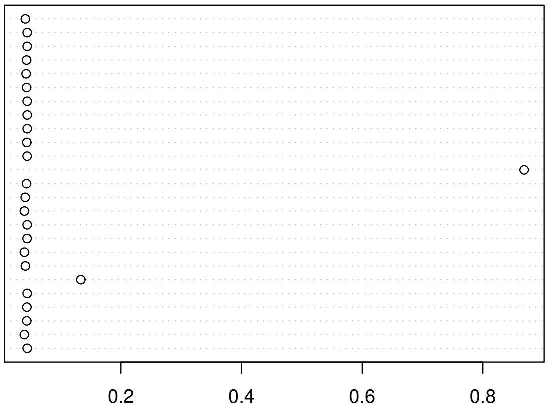

Figura 7.13: Valori sulla diagonale della matrice di proiezione.

```
 [2,]  629.7996
 [3,]  225.4648
 [4,]  217.3158
 [5,]  192.2088
 [6,] 2593.4689
 [7,]  367.5997
 [8,]  682.6168
 [9,]  200.6595
[10,]  194.3403
[11,]  690.1621
[12,]  386.4630
[13,]  249.7041
[14,] 6467.0472
[15,]  197.6414
[16,]  224.2387
[17,]  191.3411
[18,]  191.2496
[19,]  193.5858
[20,]  243.5735
[21,]  285.5444
```

```
[22,]  240.4234
[23,]  191.4920
[24,]  191.3429
[25,]  372.3155
>
```

i residui:

```
> residui<-massa-massa.prevista
> residui[1:10]
           [,1]
 [1,] -184.40115
 [2,] -206.79956
 [3,] -105.96477
 [4,] -102.31582
 [5,] -186.70877
 [6,] 2009.53110
 [7,]   51.40028
 [8,]  -27.61680
 [9,]  -85.65953
[10,] -168.74033
>
```

Un loro grafico è in figura 7.14.

```
> plot(massa.prevista,residui,xlab="Massa cerebrale prevista",
+ ylab="Residui")
>
```

Il grafico dei residui contro i valori previsti, evidenzia la presenza di alcuni residui elevati, il primo in corrispondenza dell'elefante asiatico, il secondo riferito all'uomo dove, il valore previsto è pari a 250 grammi contro i 1320 osservati. Il grafico non è del tutto soddisfacente. Calcoliamo infine l'indice di determinazione e le varianze:

```
> var.totale<-sum((massa-mean(massa))^2)/length(massa)
> var.residua<-sum(residui^2)/length(residui)
> var.spiegata<-var.totale-var.residua
> R2<-var.spiegata/var.totale> > >
```

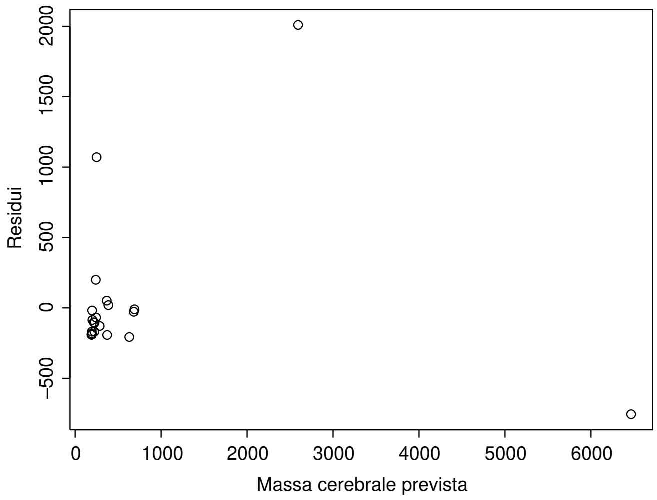

Figura 7.14: Valori previsti e residui del modello.

```
>
> var.totale
[1] 1892996
> var.residua
[1] 249223
> var.spiegata
[1] 1643773
> r2
[1] 0.8683447
>
```

Il valore di R^2 conferma tuttavia che la retta di regressione risulta adeguata.

Tuttavia come già notato, gli animali con peso corporeo piccolo sono mal rappresentati, così pure da un'analisi più attenta del grafico della massa cerebrale contro il peso corporeo si nota che anche gli animali con un peso corporeo elevato non sono ben rappresentati dalla retta di regressione. Questo è dovuto principalmente al fatto che il fenomeno analizzato varia su diversi ordini di grandezza. Per ovviare a questo problema, alla presenza di punti leva e per migliorare il grafico dei residui può essere interessante analizzare una trasformazione di entrambe le variabili su scala logaritmica in base 10.

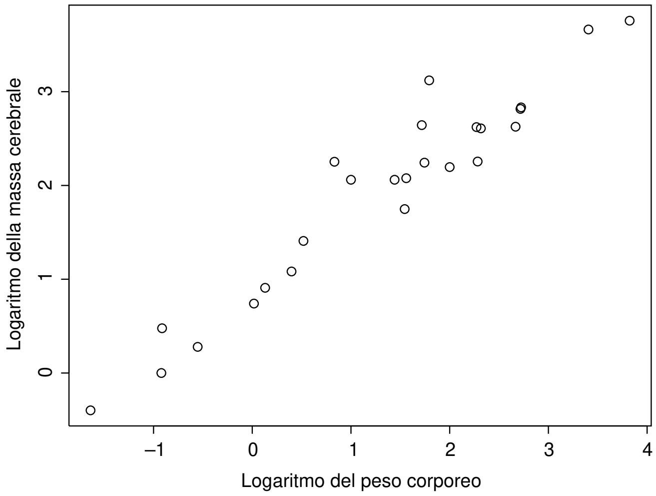

Figura 7.15: Grafico dei dati trasformati.

Calcoliamo le nuove variabili:

```
> logpeso<-log10(peso)
> logmassa<-log10(massa)
```

e rappresentiamole graficamente:

```
> plot(logpeso,logmassa,xlab="Logaritmo del peso corporeo",
+ ylab="Logaritmo della massa cerebrale")
>
```

Il grafico è riportato in figura 7.15. In questo grafico risulta evidente la possibilità di usare una retta di regressione. Ci costruiamo la matrice delle variabili esplicative:

```
> logX<-cbind(uno,logpeso)
>
```

e calcoliamo il valore dei parametri con il metodo dei minimi quadrati:

```
> logpar<-solve(t(logX)%*%logX)%*%t(logX)%*%logmassa
> logpar
```

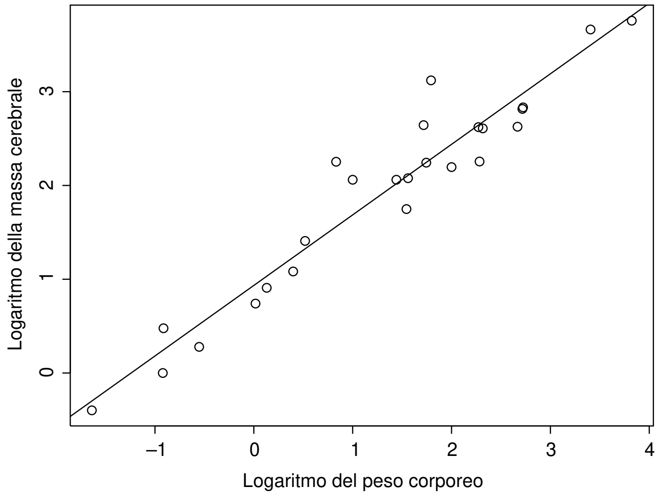

Figura 7.16: Modello ottenuto dai dati trasformati.

```
         [,1]
[1,] 0.933923
[2,] 0.752266
>
```

aggiungiamo al grafico precedente la retta di regressione:

```
> plot(logpeso,logmassa,xlab="Logaritmo del peso corporeo",
+ ylab="Logaritmo della massa cerebrale")
>
> abline(logpar[1,],logpar[2,])
>
```

Il grafico è riportato in figura 7.16. Controlliamo se vi è la presenza di punti leva calcolando la matrice di proiezione:

```
> hatlog<-logX%*%solve(t(logX)%*%logX)%*%t(logX)
> dhatlog<-diag(hatlog)
> dhatlog
[1] 0.06952839 0.07846762 0.04126019 0.04033772 0.07544587 0.13196097
[7] 0.05925276 0.08132850 0.04208910 0.05334920 0.08171968 0.06106983
```

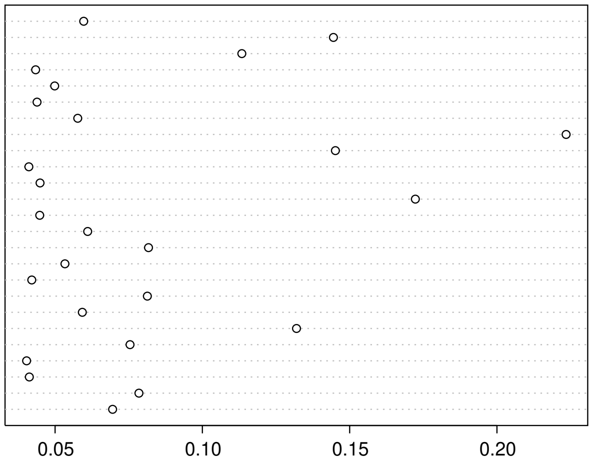

Figura 7.17: Valori sulla diagonale della matrice di proiezione basata sui dati trasformati.

```
[13] 0.04479107 0.17230524 0.04489981 0.04110237 0.14516510 0.22348134
[19] 0.05769595 0.04387401 0.04986603 0.04340263 0.11339892 0.14449092
[25] 0.05971676
>
```

Una rappresentazione grafica (in figura 7.17) può essere ottenuta come segue:

```
> dotplot(t(dhatlog))
```

Si nota come i valori associati agli elefanti si siano notevolmente ridotti. Calcoliamo i valori previsti:

```
> logmassa.prevista<-logX%*%logpar
> logmassa.prevista
            [,1]
 [1,]  1.0319687
 [2,]  2.9405570
 [3,]  2.1073876
 [4,]  2.0185793
 [5,]  0.9467366
 [6,]  3.4961626
```

```
 [7,]  2.6429521
 [8,]  2.9777075
 [9,]  1.6861890
[10,]  1.3239834
[11,]  2.9826860
[12,]  2.6761486
[13,]  2.2822782
[14,]  3.8098980
[15,]  1.5601911
[16,]  2.0954728
[17,]  0.2412224
[18,] -0.2984934
[19,]  1.2332797
[20,]  2.2460952
[21,]  2.4384549
[22,]  2.2258176
[23,]  0.5180388
[24,]  0.2466226
[25,]  2.6515728
>
```

e i residui:

```
> logres<-logmassa-logmassa.prevista
> logres[1:10]
             [,1]
 [1,] -0.123483640
 [2,] -0.314216658
 [3,] -0.030019733
 [4,]  0.042118545
 [5,] -0.206373914
 [6,]  0.166878381
 [7,] -0.020738058
 [8,] -0.161466207
 [9,]  0.374508890
[10,]  0.084256582
>
```

Il grafico (riportato in figura 7.18) che ne corrisponde evidenzia la mancanza di un qualche andamento non casuale:

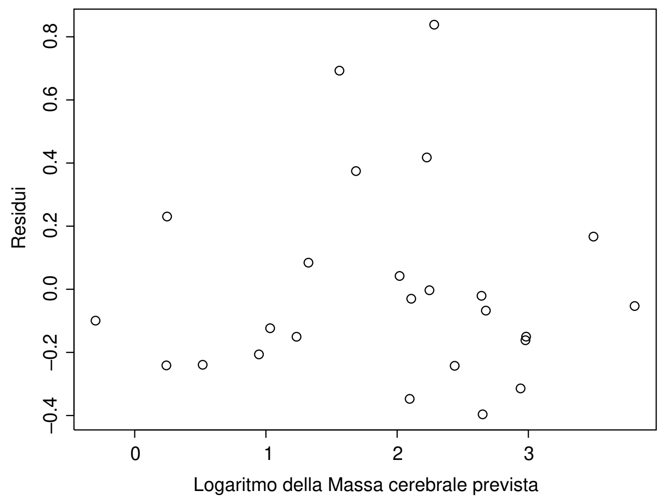

Figura 7.18: Valori previsti e residui sui dati trasformati.

```
> plot(logmassa.prevista,logres,
+ xlab="Logaritmo della Massa cerebrale prevista",
+ ylab="Residui")
>
```

Calcoliamo infine R^2 e le varianze:

```
> logvar.totale<-sum((logmassa-mean(logmassa))^2)/
+ length(logmassa)
>
> logvar.residua<-sum(logres^2)/length(logres)
> logvar.spiegata<-logvar.totale-logvar.residua
> logr2<-logvar.spiegata/logvar.totale
> logvar.totale
[1] 1.167536
> logvar.residua
[1] 0.09141816
> logvar.spiegata
[1] 1.076118
> logr2
[1] 0.9217
```

```
>
```

Si noti che la retta di regressione che abbiamo appena studiamo pari a:

$$\log_{10} massa = \gamma + \delta \log_{10} peso$$

corrisponde alla seguente relazione non lineare sulle variabili originali:

$$massa = 10^{\gamma} \times peso^{\delta}$$

da cui è immediato notare che a un peso corporeo nullo corrisponde una massa cerebrale nulla. Con questa formula possiamo calcolare i valori previsti dal modello nelle variabili originali:

```
> massa.prevista.10<-10^logpar[1,] * peso^logpar[2,]
> massa.prevista.10
[1] 10.76387 872.0814 128.052 104.3708 8.8457 3134.45
[7] 439.4931 949.9647 48.5499 21.08547 960.91 474.404
[13] 191.548 6455.026 36.3237 124.5870 1.7426 0.50292
[19] 17.1111 176.2362 274.444 168.1967 3.2963 1.76450
[25]  448.30415
>
```

da cui si nota come siano migliarati i valoti previsti per gli animali con un peso notevole e sicuramente quelle con un peso corporeo molto piccolo, in particolare il valore previsto per il maiale della guinea è pari a 8.84 grammi contro i 192.2 della retta di regressione. Il grafico di questa regressione (figura 7.19) si ottiene come segue:

```
> linea<-seq(min(peso),max(peso),by=100)
> plot(peso,massa,xlab="Peso corporeo",
+ ylab="Massa cerebrale",
+ xlim=c(0,600),ylim=c(0,1500))
>
> lines(linea,10^logpar[1,] * linea^logpar[2,])
>
```

Si noti infine che la somma dei residui dalla seconda regressione espressi nelle variabili originarie non è zero:

```
> residui.10<-massa-massa.prevista.10
> sum(residui.10)
[1] 760.027
>
```

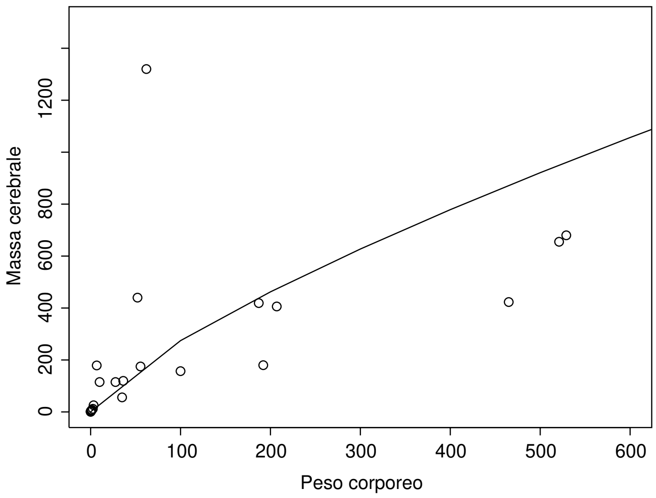

Figura 7.19: Modello calcolato sui dati trasformati riportato sui dati originali.

Parte II

Alcuni approfondimenti

Capitolo 8

Ancora sulla regressione

8.1 Sigarette

Il primo insieme di dati che utilizziamo riguarda le caratteristiche di alcuni tipi di sigarette. Carichiamo i dati contenuti nel *file* `"sigar.txt"`, costituito da cinque colonne che rappresentano rispettivamente la marca (V1), il condensato (contenuto di catrame in mg, V2), il peso (gr, V3) ed il contenuto di monossido di carbonio (mg, V5) per 25 diverse marche di sigarette.

```
> sigar<-read.table("sigar.txt")
>
> sigar
               V1   V2   V3     V4   V5
1          Alpine 14.1 0.86 0.9853 13.6
2    Benson&Hedges 16.0 1.06 1.0938 16.6
3       BullDurham 29.8 2.03 1.1650 23.5
4      CamelLights  8.0 0.67 0.9280 10.2
5         Carlton  4.1 0.40 0.9462  5.4
6     Chesterfield 15.0 1.04 0.8885 15.0
7     GoldenLights  8.8 0.76 1.0267  9.0
8            Kent 12.4 0.95 0.9225 12.3
9            Kool 16.6 1.12 0.9372 16.3
10            L&M 14.9 1.02 0.8858 15.4
11     LarkLights 13.7 1.01 0.9643 13.0
12       Marlboro 15.1 0.90 0.9316 14.4
13          Merit  7.8 0.57 0.9705 10.0
```

```
14       MultiFilter 11.4 0.78 1.1240 10.2
15     NewportLights  9.0 0.74 0.8517  9.5
16               Now  1.0 0.13 0.7851  1.5
17           OldGold 17.0 1.26 0.9186 18.5
18     PallMallLight 12.8 1.08 1.0395 12.6
19           Raleigh 15.8 0.96 0.9573 17.5
20        SalemUltra  4.5 0.42 0.9106  4.9
21          Tareyton 14.5 1.01 1.0070 15.9
22              True  7.3 0.61 0.9806  8.5
23 ViceroyRichLight  8.6 0.69 0.9693 10.6
24     VirginiaSlims 15.2 1.02 0.9496 13.9
25     WinstonLights 12.0 0.82 1.1184 14.9
>
```

Vogliamo verificare se esiste una relazione tra il monossido di carbonio (variabile dipendente) ed il condensato (variabile esplicativa). Assegniamo i dati sul condensato e quelli sul monossido di carbonio a due vettori distinti.

```
> condens<-sigar[,2]
> monoss<-sigar[,5]
>
```

Per dare un primo sguardo alla relazione tra le due variabili, costruiamo un grafico che rappresenti la nuvola di punti osservati.

```
> plot(condens,monoss,xlab="Condensato",ylab="Monossido",
+ main="Grafico Monossido Condensato")
>
```

Il grafico è in figura 8.1. Come si vede, tra le variabili c'è una relazione lineare abbastanza forte. Risulta inoltre evidente che un'osservazione è al di fuori della nuvola in cui stanno la maggior parte dei punti.

Calcoliamo i parametri della retta di regressione:

$$monossido = a + b\ condensato$$

È noto che, se chiamiamo x la variabile esplicativa ed y la variabile dipendente, i parametri ottenuti con il principio dei minimi quadrati sono dati da:

$$\hat{b} = \frac{\mathrm{Cov}(x,y)}{\mathrm{Var}(x)} \qquad \text{e} \qquad \hat{a} = \mathrm{M}(y) - b\ M(x) \tag{8.1}$$

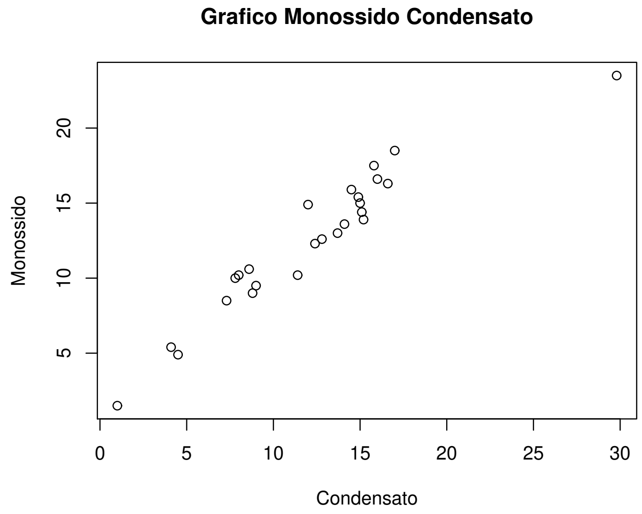

Figura 8.1: Grafico del Monossido rispetto al condensato.

Calcoliamo innanzitutto le medie delle due variabili:

```
> media.condens<-mean(condens)
> media.condens
[1] 12.216
>
> media.monoss<-mean(monoss)
> media.monoss
[1] 12.528
>
```

Calcoliamo ora le varianze:

```
> n<-length(condens)
>
> var.condens<-sum(condens^2)/n - media.condens^2
> var.condens
[1] 30.81734
>
> var.monoss<-sum(monoss^2)/n - media.monoss^2
> var.monoss
```

```
[1] 21.56602
>
```

L'ultima cosa che rimane da calcolare è la covarianza:

```
> covar<-sum((condens-media.condens)*(monoss-media.monoss))/n
> covar
[1] 24.68395
>
```

Dal valore della covarianza calcoliamo rapidamente il coefficiente di correlazione tra il monossido e il condensato:

```
> correlazione<-covar/sqrt(var.condens * var.monoss)
> correlazione
[1] 0.9574853
>
```

Con questo valore elevato possiamo aspettarci un buon modello. Facendo adesso riferimento alla 8.1, troviamo:

```
> b<-covar/var.condens
> b
[1] 0.800976
>
> a<-media.monoss - b * media.condens
> a
[1] 2.743278
>
```

La funzione `abline(a,b)` aggiunge al grafico corrente una retta con intercetta `a` e coefficiente angolare `b`.

```
> abline(a,b)
>
```

Il risultato è mostrato in figura 8.2.

Calcoliamo adesso i residui della regressione, cioè la differenza tra i valori osservati ed i valori previsti dalla retta.

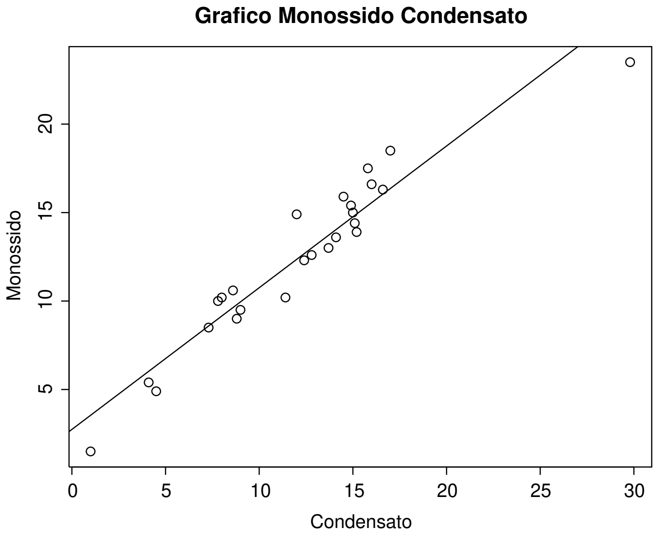

Figura 8.2: Retta del monossido sul condensato.

```
> prev.monoss<-a + b * condens
> residui<-monoss - prev.monoss
>
> residui[1:5]
[1] -0.4370387  1.0411069 -3.1123615  1.0489147 -0.6272790
>
```

Come si vede, i residui assumono valori attorno allo zero. Il residuo corrispondente alla terza osservazione è, in valore assoluto, molto elevato. Questo residuo è riferito al punto che avevamo già visto essere al di fuori della nuvola. Tracciamo ora il grafico dei residui contro la variabile esplicativa.

```
> plot(condens,residui,xlab="Condensato",ylab="Residui",
+ main="Grafico Condensato Residui")
```

Il grafico è riportato in figura 8.3.

Una procedura alternativa è quella di utilizzare una funzione specifica di R chiamata `lm()` (*linear model*). Vediamo come si usa:

```
> lm(monoss~condens)
```

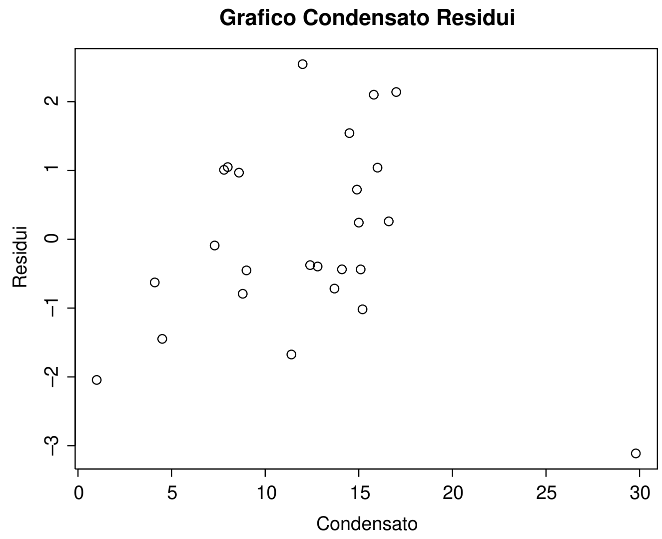

Figura 8.3: Residui della regressione lineare contro i valori della variabile esplicativa.

```
Call:
lm(formula = monoss ~ condens)

Coefficients:
(Intercept)      condens
      2.743        0.801

>
```

(Per chi usa una tastiera italiana per ottenere il simbolo ˜ deve utilizzare la combinazione `ALT`+126).

Quindi la funzione `lm()` richiede la variabile dipendente, nel nostro caso il monossido seguita dal simbolo ˜ e dalla variabile esplicativa.

I valori che abbiamo ottenuto coincidono con i valori dei parametri che avevamo già trovato.

Assegnando il risultato a una variabile nel modo seguente possiamo utilizzare ulteriormente i valori ottenuti:

```
> modello<-lm(monoss~condens)
>
```

Possiamo ottenere il valore dei parametri con la funzione `coefficients()`:

```
> coeff.modello<-coefficients(modello)
> coeff.modello
(Intercept)     condens
   2.743278    0.800976
>
```

I valori previsti possono essere ottenuti con la funzione `fitted.values()`:

```
> prev.monoss<-fitted.values(modello)
>
```

mentre i residui (`residuals()`):

```
> residui<-residuals(modello)
>
```

Infine con la funzione `summary()` abbiamo:

```
> summary(modello)

Call:
lm(formula = monoss ~ condens)

Residuals:
    Min      1Q  Median      3Q     Max
-3.1124 -0.7166 -0.3754  1.0091  2.5450

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.74328    0.67521   4.063 0.000481 ***
condens      0.80098    0.05032  15.918 6.55e-14 ***
---
Signif. codes: 0 ‘***’  0.001 ‘**’  0.01 ‘*’  0.05 ‘.’  0.1 ‘ ’ 1

Residual standard error: 1.397 on 23 degrees of freedom
Multiple R-Squared: 0.9168,     Adjusted R-squared: 0.9132
F-statistic: 253.4 on 1 and 23 degrees of freedom, p-value: 6.55e-14

>
```

Vediamo in dettaglio le informazioni fornite da questa funzione. La sezione `Call:` contiene la funzione del modello utilizzato, in `Residuals:` sono riportati alcuni quantili dei residui: il valore minimo, il primo quartile, la mediana, il terzo quartile e il valore massimo. Nella sezione `Coefficients:` troviamo il valore del parametro (*Estimate*), il corrispondente errore standard (*Std. Error*), il valore del test t (*t value*) calcolato sotto l'ipotesi nulla che il valore del parametro sia nullo e il valore p ($Pr(>|t|)$) che è il massimo valore dell'errore di primo tipo che porta ad accettare l'ipotesi nulla. Infine, nell'ultima colonna viene riportato con simboli il livello di significatività del test. Sotto la voce `Residual standard error:` abbiano una stima della deviazione standard del termine di errore del modello, e i gradi di libertà. In `Multiple R-Squared:` abbiamo il calcolo del coefficiente di determinazione R^2:

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum_{i=1}^n r_i^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

dove SSE è la devianza residua cioè del termine di errore, mentre SST è la devianza della variabile dipendente. In `Adjusted R-squared:` il valore di R^2 "aggiustato" con i gradi di libertà:

$$\bar{R}^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

dove n è il numero di osservazioni e p il numero di variabili esplicative (esclusa l'intercetta).

Infine abbiamo la statisica F che ci fornisce una valutazione complessiva sulla significatività del modello rispetto ad un modello che contenga la sola intercetta.

Inoltre, associando una variabile alla funzione `summary()`ad esempio:

```
> summary.modello<-summary(modello)
>
```

possiamo riutilizzare i valori contenuti in `summary()` attraverso le seguenti istruzioni:

`summary.modello$coefficients` ci fornisce i valori dei parametri, gli errori standard, il valore del test t e il valore p;

`summary.modello$sigma` l'errore standard per il termine d'errore (la radice quadrata della stima della varianza);

`summary.modello$df` i gradi di libertà

`summary.modello$fstatistic` il valore della statistica F;

`summary.modello$r.squared` il valore del coefficiente di determinazione R^2;

`summary.modello$adj.r.squared` il valore del coefficiente di determinazione $\bar{R}^2$ "aggiustato"

Dai valori molto piccoli di `Pr(>|t|)` (dalla presenza di ***) possiamo co-cludere che entrambe le variabili esplicative hanno un coeffieciente di regressione diverso da zero. L'elevato valore di R^2 e della statistica F confermano che il modello scelto è un buon modello.

L'analisi dei quantili dei residui evidenzia la presenza i una asimmetria e di un valore minimo particolarmente elevato (in valore assoluto).

8.2 Grasso

Nell'esempio che presentiamo analizziamo la relazione che esiste tra l'età e la percentuale di grasso presente nel corpo umano. A tale scopo è stata misurata l'età, la percentuale di grasso e il genere di 24 individui adulti. I dati sono presenti nel *file*: `"grasso.txt"`.

```
> dati<-read.table("grasso.txt")
> dati
   V1   V2 V3
1  23  8.5  m
2  23 21.9  f
3  24 12.2  m
4  25 24.0  f
5  26 12.5  m
6  27  8.8  m
7  27 12.8  m
8  30 14.1  m
9  32 12.9  m
10 35 31.2  f
11 39 33.4  f
12 41 27.9  f
```

```
13 45 23.4  m
14 49 30.2  f
15 50 37.1  f
16 53 42.7  f
17 53 48.0  f
18 54 35.1  f
19 56 38.5  f
20 57 36.3  f
21 58 39.0  f
22 58 39.8  f
23 60 47.1  f
24 61 40.5  f
>
```

Definiamo le tre variabili:

```
> eta<-dati[,1]
> grasso<-dati[,2]
> sesso<-dati[,3]
>
```

Costruiamo il diagramma di dispersione tra l'età e la percentuale di grasso (figura 8.4).

```
> plot(eta,grasso,xlab="Eta'",
+ ylab="Percentuale di grasso corporeo")
>
```

La relazione è abbastanza lineare e quindi cercheremo di adattare un modello di regressione lineare. Si badi però che un tale modello deve essere usata con cautela a fini previsivi: verosimilmente la percentuale di grasso ha un limite fisiologico. Allora, se si usa un modello di regressione lineare e si estrapola la percentuale di grasso rispetto a valori elevati (più grandi di quelli osservati) dell'età, si corre il rischio di ottenere una valore previsto che supera tale limite fisiologico.

Il modello che vogliamo utilizzare è:

$$grasso = a + b\ eta + \varepsilon$$

quindi:

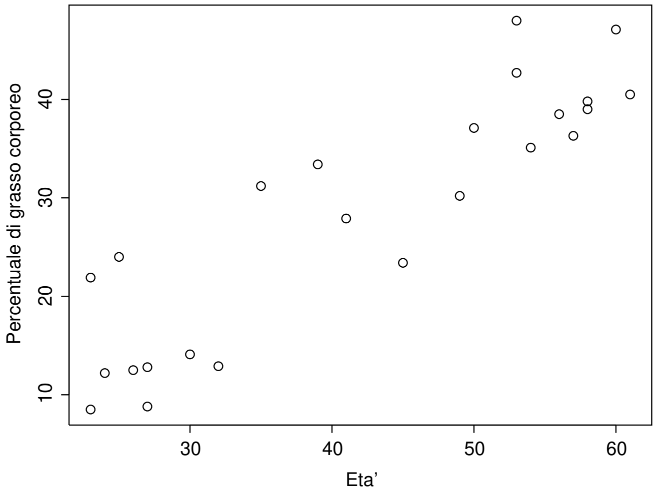

Figura 8.4: Grafico tra l'età e la percentuale di grasso.

```
> modello<-lm(grasso~eta)
>
```

assegnamo ad un vettore il valore di a e b ottenuto con il princio dei minimi quadrati

```
> coeff.modello<-coefficients(modello)
> coeff.modello
(Intercept)         eta
 -5.9029103   0.8146818
>
```

rappresentiamo la retta ottenuta (figura 8.5):

```
> plot(eta,grasso,xlab="Eta'",ylab="Percentuale di grasso corporeo")
> abline(coeff.modello)
>
```

Per alcuni individui sembra esserci una certa discrepanza tra osservazione e retta di regressione, ma complessivamente l'adattamento è piuttosto buono. Verifichiamo la bontà di adattamento del modello osservando i risultati forniti dalla funzione `summary()`:

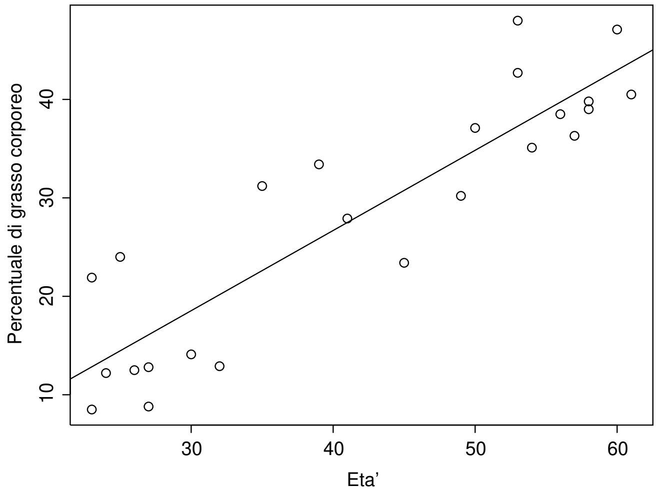

Figura 8.5: Grafico tra l'età e la percentuale di grasso, ed il modello stimato.

```
> summary(modello)
Call:
lm(formula = grasso ~ eta)

Residuals:
   Min     1Q Median     3Q    Max
-7.358 -3.921 -1.949  4.448 10.725

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.90291    3.85314  -1.532    0.140
eta          0.81468    0.08741   9.320 4.28e-09 ***
---
Signif. codes: 0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 5.844 on 22 degrees of freedom
Multiple R-Squared: 0.7979,     Adjusted R-squared: 0.7887
F-statistic: 86.87 on 1 and 22 degrees of freedom, p-value: 4.275e-09

>
```

Notiamo che nel modello l'intercetta potrebbe essere rimossa. L'adattamento è buono e il coefficiente di determinazione è elevato. Poniamo il valore di R^2 in una nuova variabile che ci servirà successivamente:

```
> R2.modello<-summary(modello)$r.squared
> R2.modello
[1] 0.797926
>
```

infine calcoliamo i valori previsti e i residui del modello:

```
> prev.modello<-fitted.values(modello)
> residui<-residuals(modello)
>
```

Conviene rappresentare separatamente i residui che si riferiscono ad individui maschi da quelli delle femmine:

```
> plot(eta,residui,type="n",xlab="Eta'",ylab="Residui")
> points(eta[sesso=="m"],residui[sesso=="m"],pch="m")
> points(eta[sesso=="f"],residui[sesso=="f"],pch="f")
>
```

Il grafico è in figura 8.6. In generale, non sono però identificabili preoccupanti *pattern* di variazione. Tuttavia, appare chiaro che i residui per i maschi sono più piccoli di quelli per le femmine. Appare allora sensato adattare un modello in cui si possa distinguere tra le osservazioni che si riferiscono ai maschi e quelle che si riferiscono alle femmine. A questo scopo costruiamo una nuova variabile (dicotomica) che assume valore 1 per le femmine e 0 per i maschi nel seguente modo:

```
> sesso.bin<-(sesso=="f")*1
> sesso.bin
[1] 0 1 0 1 0 0 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1
>
```

e proviamo a stimare il seguente modello di regressione lineare:

$$grasso = \alpha + \beta eta + \gamma sesso.bin + \varepsilon'$$

cioè:

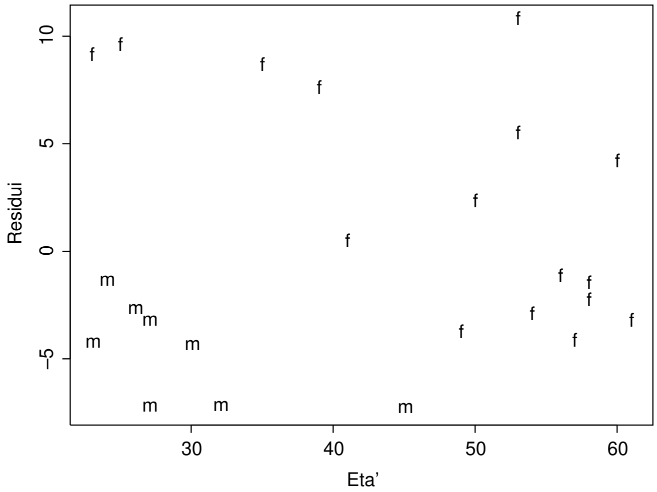

Figura 8.6: Grafico tra l'età, la percentuale di grasso e il sesso.

```
> modello.bin<-lm(grasso~eta+sesso.bin)
> coeff.modello.bin<-coefficients(modello.bin)
> prev.modello.bin<-fitted.values(modello.bin)
> residui.modello.bin<-residuals(modello.bin)
>
```

Vediamo graficamente come sono cambiati i residui del nuovo modello (identificati dal simbolo +) rispetto ai precedenti (simbolo ∘):

```
> plot(eta,residui,xlab="Eta'",ylab="Residui")
> points(eta,residui.modello.bin,pch="+")
>
```

Il grafico è in figura 8.7.

Il modello ha le seguenti caratteristiche:

```
> summary(modello.bin)

Call:
lm(formula = grasso ~ eta + sesso.bin)
```

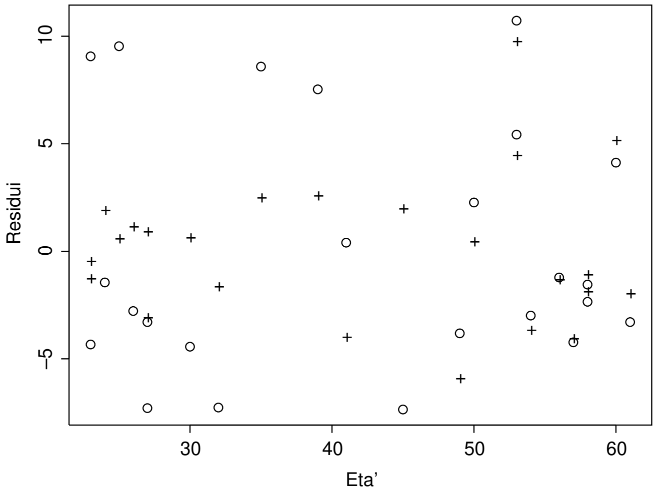

Figura 8.7: Grafico tra l'età e i residui di due diversi modelli.

```
Residuals:
     Min       1Q   Median       3Q      Max
-5.99076 -1.97690 -0.07392  1.85000  9.69185

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.33342    2.43682  -0.958    0.349
eta          0.52935    0.07113   7.442 2.57e-07 ***
sesso.bin   12.58615    2.05923   6.112 4.59e-06 ***
---
Signif. codes: 0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 3.588 on 21 degrees of freedom
Multiple R-Squared: 0.9273,     Adjusted R-squared: 0.9204
F-statistic: 133.9 on 2 and 21 degrees of freedom, p-value: 1.115e-12
>
```

La variabile *sesso* risulta sicuramente importante nel descrivere la percentuale di grasso, il coefficiente di determinazione è aumentato ulteriormente.

Per evidenziare l'importanza della variabile sesso nel descrivere la percentuale di grasso, possiamo calcolare il coefficiente di correlazione parziale tra la percentuale di grasso e sesso, al netto dell'influenza lineare della variabile età:

```
> R2.modello.bin<-summary(modello.bin)$r.squared
> R2.modello.bin
[1] 0.9272833
>
> corr.parz<-sqrt((R2.modello.bin-R2.modello)/(1-R2.modello))
> corr.parz
>
[1] 0.8000925
>
```

da cui risulta che la variabile *sesso* risulta importante nel spiegare la variabilità della percentuale di grasso.

La figura 8.8 mostra le rette di regressione per i maschi (le osservazioni per i quali sono indicate con il simbolo $\circ$) e per le femmine (indicate da $\triangle$), che differiscono per l'intercetta, come viene suggerito dall'ultimo modello calcolato:

```
plot(eta,grasso,xlab="Eta'",ylab="Percentuale di grasso",
type="n")
points(eta[sesso=="m"],grasso[sesso=="m"])
points(eta[sesso=="f"],grasso[sesso=="f"],pch=2)
abline(coeff.modello.bin[1],coeff.modello.bin[2])
abline(coeff.modello.bin[1]+coeff.modello.bin[3],
coeff.modello.bin[2],lty=2)
```

Si noti come la differenza tra le due intercette è circa 12.586, mentre la differenza tra la percentuale di grasso media per i maschi e per le femmine è:

```
> mean(grasso[sesso=="f"])-mean(grasso[sesso=="m"])
[1] 22.64375
>
```

Questo discende dal fatto che facendo semplicemente la differenza tra le medie per i due sessi, ignoriamo l'influenza dell'età. Il valore di γ ci suggerisce che,

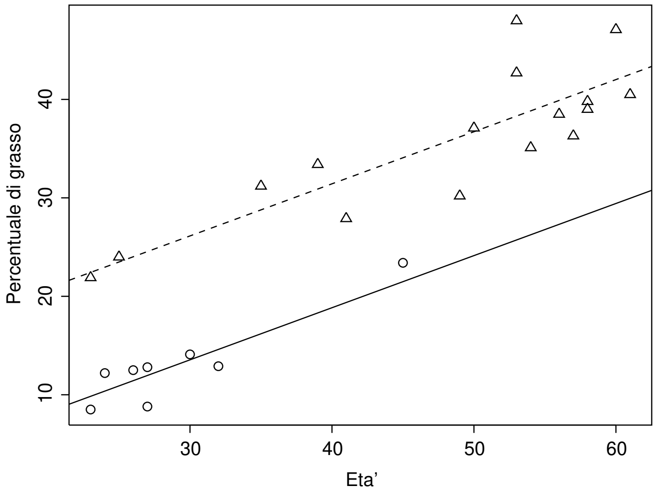

Figura 8.8: Percentuale di grasso corporeo rispetto all'età per maschi (○) e femmine (△), con la retta di regressione per i maschi (linea continua) e per le femmine (linea tratteggiata).

a parità di età, la differenza media tra la percentuale di grasso per i maschi e quella per le femmine è di circa 12.586.

Infine è facile notare dal grafico in figura 8.8 che abbiamo posto nel nostro modello i coefficienti angolari uguali per i maschi e per le femmine. È opportuno verificare questa ipotesi considerando un modello del tipo:

$$grasso = \Phi + \Omega eta + \Gamma sesso.bin + \Delta sesso.bin \times eta + \varepsilon''$$

Infatti per questo modello otteniamo che per i maschi abbiamo:

$$grasso_m = \Phi + \Omega eta_m + \varepsilon''$$

mentre per le femmine:

$$grasso_f = (\Phi + \Gamma) + (\Omega + \Delta) eta_f + \varepsilon''$$

e quindi basterà verificare se il valore del coefficiente Δ che misura la differenza tra i coefficienti angolari di maschi e femmine può essere considerato nullo.

```
> summary(lm(grasso~eta*sesso.bin))

Call:
lm(formula = grasso ~ eta * sesso.bin)

Residuals:
    Min      1Q  Median      3Q     Max
-5.9818 -1.8968 -0.2976  1.5479  9.7488

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   -4.57882    5.92055  -0.773  0.44834
eta            0.60611    0.19752   3.069  0.00606 **
sesso.bin     15.40961    7.07566   2.178  0.04156  *
eta.sesso.bin -0.08875    0.21237  -0.418  0.68048
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.661 on 20 degrees of freedom
Multiple R-Squared: 0.9279,     Adjusted R-squared: 0.9171
F-statistic: 85.81 on 3 and 20 degrees of freedom, p-value: 1.355e-11
```

Dal valore del test t possiamo concludere che $\Delta = 0$ e che quindi i coefficienti angolari possono essere considerati uguali.

8.3 Energia eolica

Nell'esempio che segue illustriamo un caso di regressione intrinsecamente lineare. Con regressione intrinsecamente lineare ci si riferisce al caso in cui la variabile dipendente y non è legata alla variabile esplicativa x in maniera lineare, ma si può scrivere:

$$g(y) = a + bh(x)$$

per opportune funzioni di trasformazione $g(\cdot)$ e $h(\cdot)$.

L'insieme di dati che ci accingiamo a studiare la variabile dipendente è l'intensità della corrente elettrica (misurata in Ampère) prodotta da un mulino a vento e la variabile esplicativa è la velocità del vento. I dati sono nel *file* `"vento.txt"`.

```
> vento<-read.table("vento.txt")
>
> vento
      V1    V2
1   5.00 1.582
2   6.00 1.822
3   3.40 1.057
4   2.70 0.500
5  10.00 2.236
6   9.70 2.386
7   9.55 2.294
8   3.05 0.558
9   8.15 2.166
10  6.20 1.866
11  2.90 0.653
12  6.35 1.930
13  4.60 1.562
14  5.80 1.737
15  7.40 2.088
16  3.60 1.137
17  7.85 2.179
18  8.80 2.112
19  7.00 1.800
20  5.45 1.501
21  9.10 2.303
22 10.20 2.310
23  4.10 1.194
24  3.95 1.144
25  2.45 0.123
>
```

Salviamo su due vettori distinti la velocità e la corrente prodotta.

```
> vel<-vento[,1]
> dc<-vento[,2]
>
```

Per renderci conto del tipo di relazione tra le due variabili, costruiamo un grafico dell'intensità della corrente prodotta contro la velocità del vento.

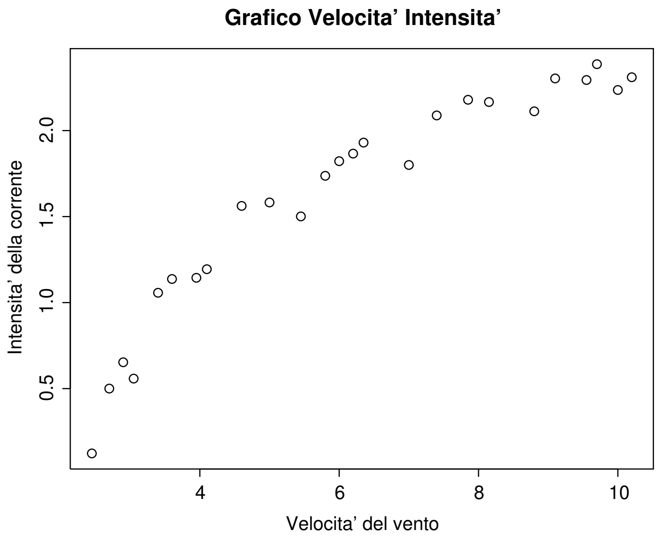

Figura 8.9: Grafico dell'Intensità della corrente prodotta rispetto alla velocità del vento.

```
> plot(vel,dc,xlab="Velocita' del vento",
+ ylab="Intensita' della corrente",
+ main="Grafico Velocita' Intensita'")
>
```

Il grafico è riportato nella figura 8.9.

La figura mostra che la relazione tra le due variabili non è lineare. Un'altra cosa da notare è che, al crescere della velocità del vento, l'intensità della corrente prodotta sembra raggiungere un asintoto.

Vediamo cosa succede se interpoliamo i dati con una retta di regressione nella forma:

$$\text{corrente} = a + b \text{ velocià} + \varepsilon \qquad (8.2)$$

Le istruzioni sono:

```
> modello.lin<-lm(dc~vel)
> summary(modello.lin)

Call:
lm(formula = dc ~ vel)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-0.59869 -0.14099  0.06059  0.17262  0.32184

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.13088    0.12599   1.039     0.31
vel          0.24115    0.01905  12.659 7.55e-12 ***
---
Signif. codes:  0  `***'  0.001  `**'  0.01  `*'  0.05  `.'  0.1  ` '  1

Residual standard error: 0.2361 on 23 degrees of freedom
Multiple R-Squared: 0.8745,     Adjusted R-squared: 0.869
F-statistic: 160.3 on 1 and 23 degrees of freedom,      p-value: 7.546e-12

>
```

L'indice di determinazione R^2 è il rapporto tra la varianza spiegata e la varianza totale, dove quest'ultima è la varianza della variabile dipendente, noi possiamo ottenerne il valore con:

```
> R2<-summary(modello.lin)$r.squared
> R2
[1] 0.8744932
>
```

Si noti che nel caso di regressione lineare semplice l'indice di determinazione R^2 coincide con il coefficiente di correlazione al quadrato:

```
> r<-cor(dc,vel)
> r^2
[1] 0.8744932
>
```

La retta di regressione spiega l'87% della varianza totale. Calcoliamo ora i residui e ne tracciamo un grafico contro la variabile esplicativa.

```
> res.modello.lin<-residuals(modello.lin)
>
```

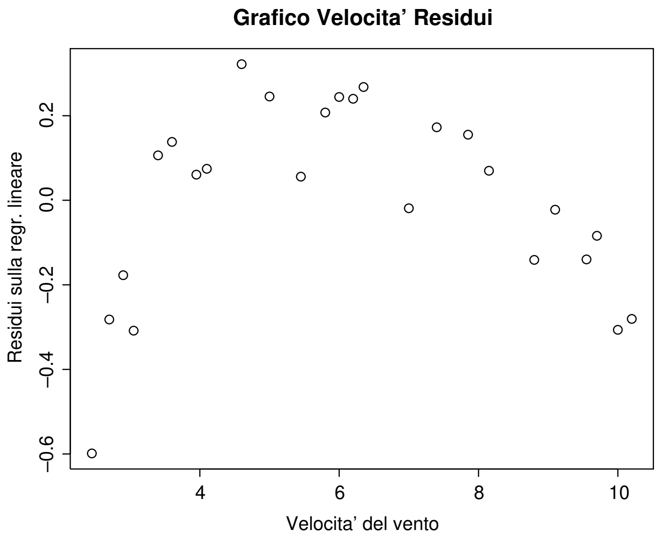

Figura 8.10: Grafico dei residui per il modello lineare rispetto alla velocità del vento.

```
> plot(vel,res.modello.lin,xlab="Velocita' del vento",
+ ylab="Residui sulla regr. lineare",
+ main="Grafico Velocita' Residui")
>
```

Il grafico è mostrato nella figura 8.10. I residui hanno un andamento niente affatto "casuale" e mettono in evidenza che la relazione tra l'intensità della corrente prodotta e la velocità del vento non è lineare. Questo ci spinge a utilizzare un modello quadratico:

$$\text{corrente} = \alpha + \beta \text{ velocità} + \gamma \text{ (velocità)}^2 + \varepsilon' \tag{8.3}$$

Per calcolare i parametri di questo modello utilizziamo nuovamente la funzione `lm()`:

```
> vel2<-vel^2
> modello.qua<-lm(dc~vel+vel2)
> summary(modello.qua)

Call:
```

```
lm(formula = dc ~ vel + vel2)

Residuals:
     Min       1Q   Median       3Q      Max
-0.26347 -0.02537  0.01263  0.03908  0.19903

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.155898   0.174650  -6.618 1.18e-06 ***
vel          0.722936   0.061425  11.769 5.77e-11 ***
vel2        -0.038121   0.004797  -7.947 6.59e-08 ***
---
Signif. codes:  0  `***'  0.001  `**'  0.01  `*'  0.05  `.'  0.1  ` '  1

Residual standard error: 0.1227 on 22 degrees of freedom
Multiple R-Squared: 0.9676,     Adjusted R-squared: 0.9646
F-statistic: 328.3 on 2 and 22 degrees of freedom,      p-value:     0

>
```

Costruiamo un grafico che mostri come il modello 8.3 interpola i dati e come prevede la corrente generata per valori della velocità del vento maggiori di quelli osservati.

```
> coeff.modello.qua<-coefficients(modello.qua)
>
> vettx<-seq(2,16,by=0.1)
>
> plot(vel,dc,xlab="Velocita' del vento",
+ ylab="Intensita' della corrente",
+ main="Grafico Velocita' Intensita'")
>
> lines(vettx,coeff.modello.qua[1]+coeff.modello.qua[2]*
+ vettx+coeff.modello.qua[3]*vettx^2)
>
```

Il grafico è riportato nella figura 8.11. La figura mostra che il modello quadratico interpola bene i dati osservati, ma non tiene conto della tendenza

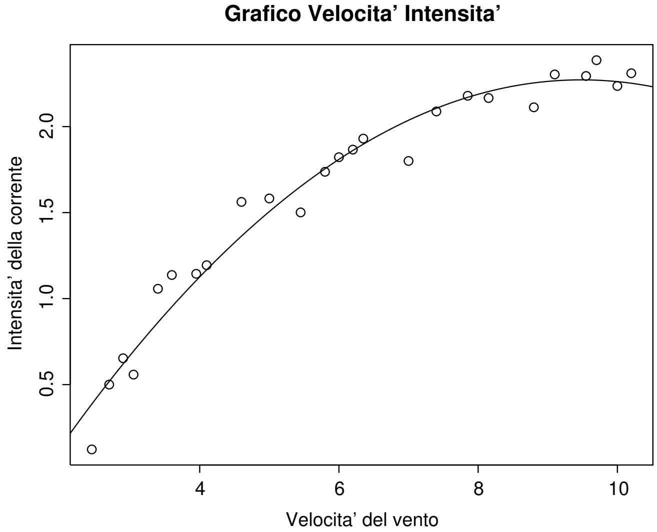

Figura 8.11: Grafico dei valori previsti (linea continua) dal modello di regressione quadratico.

dell'intensità della corrente generata a raggiungere un asintoto. Il modello, prevede che al crescere della velocità l'intensità della corrente generata decresca drasticamente. Questo ci porta a rifiutare anche questo modello.

Un modello che porta al raggiungimento di un asintoto è il seguente:

$$\text{corrente} = a + b\,\frac{1}{\text{velocità}} + \varepsilon'' \tag{8.4}$$

che è lineare nel reciproco della velocità. Calcoliamo il reciproco della velocità e tracciamo un grafico che ne rappresenti la relazione con l'intensità della corrente prodotta.

```
> recvel<-1/vel
>
> plot(recvel,dc,xlab="Reciproco della velocita' del vento",
+ ylab="Intensita' della corrente",
+ main="Grafico Reciproco velocita' Intensita'")
>
```

Il grafico è mostrato in figura 8.12, dal quale si vede come la relazione tra l'intensità della corrente prodotta e il reciproco della velocità del vento sia lineare. Calcoliamo con il solito procedimento i parametri del modello 8.3.

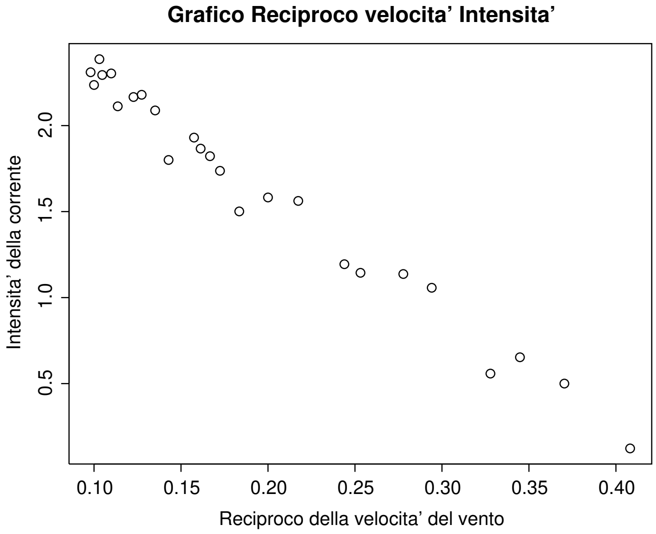

Figura 8.12: Grafico dell'intensità della corrente prodotta rispetto al reciproco della velocità del vento.

```
> modello.rec<-lm(dc~recvel)
> summary(modello.rec)
>
Call:
lm(formula = dc ~ recvel)

Residuals:
     Min       1Q   Median       3Q      Max
-0.20547 -0.04941  0.01100  0.08352  0.12204

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.9789     0.0449   66.34   <2e-16 ***
recvel       -6.9345     0.2064  -33.59   <2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 0.09417 on 23 degrees of freedom
Multiple R-Squared:  0.98,      Adjusted R-squared: 0.9792
```

```
F-statistic:  1128 on 1 and 23 degrees of freedom,  p-value: 0
```

Calcoliamo R^2 per il nuovo modello ricordando che per la regressione lineare semplice $R^2 = \rho^2(dc, recvel)$:

```
> R2.rec<-summary(modello.rec)$r.squared
> R2.rec
[1] 0.9800249
>
```

Il valore risultante è molto elevato e quindi l'adattamento del modello è buono.

Tracciamo infine un grafico dei residui per il modello 8.3 rispetto alla variabile esplicativa di quel modello, cioè il reciproco della velocità del vento.

```
> res.modello.rec<-residuals(modello.rec)
>
> plot(recvel,res.modello.rec,xlab="Reciproco della velocita' del vento",
+ ylab="Residui",main="Grafico Reciproco velocita' Residui")
>
```

Il grafico dei resiudi è presente in figura 8.13. Questa volta i residui non sembrano avere alcun particolare comportamento e siamo dunque soddisfatti.

Con le istruzioni che seguono viene costruito il grafico in figura 8.14, che mostra l'intensità della corrente elettrica prodotta prevista per i valori della velocità del vento più grandi di quelli osservati:

```
> vettx<-seq(2,16,by=0.1)
>
> plot(vel,dc,xlab="Velocita' del vento",
+ ylab="Intensita' della corrente",
+ main="Grafico Reciproco velocita' Intensita'",
+ xlim=c(0,16),ylim=c(0,3))
>
> lines(vettx,coeff.modello.rec[1]+
+ coeff.modello.rec[2]*1/vettx)
>
```

In questo caso è evidente che il modello basato sul reciproco della velocità rispetta la tendenza dell'intensità della corrente elettrica generata a raggiungere un asintoto, proprio come volevamo accadesse.

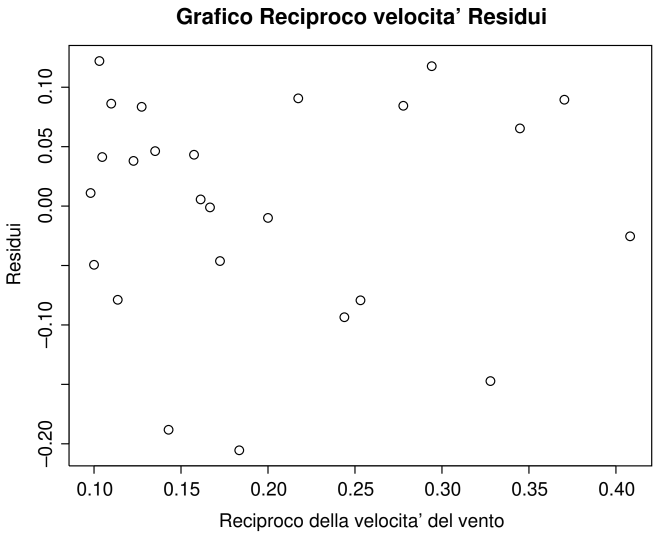

Figura 8.13: Grafico dei residui per il modello lineare nel reciproco della velocità del vento.

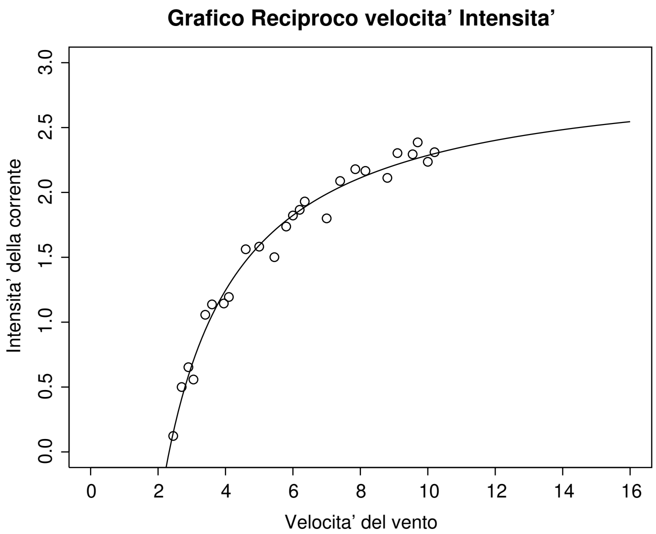

Figura 8.14: Valori previsti dal modello 8.3 per grandi velocità del vento.

8.4 Massa cerebrale

Con questo insieme di dati studiamo la relazione esistente tra il peso corporeo medio di un animale adulto e la sua massa cerebrale. Si vuole capire se per governare un peso corporeo elevato sia necessaria una maggiore massa cerebrale. L'insieme di dati riguarda, il nome dell'animale, il peso corporeo in chilogrammi e la massa cerebrale in grammi. Con il comando `read.table` leggiamo il *file* di dati `"animali.txt"`:

```
> animali<-read.table("animali.txt")
> animali
                    V1       V2     V3
1  Castoro_di_montagna    1.350    8.1
2                Mucca  465.000  423.0
3          Lupo_grigio   36.300  119.5
4                Capra   27.660  115.0
5      Maiale_di_guinea    1.040    5.5
6    Elefante_asiatico 2547.000 4603.0
7                Asino  187.000  419.0
8              Cavallo  521.000  655.0
9     Scimmia_africana   10.000  115.0
10               Gatto    3.300   25.6
11             Giraffe  529.000  680.0
12             Gorilla  207.000  406.0
13                Uomo   62.000 1320.0
14   Elefante_africano 6654.000 5712.0
15     Scimmia_indiana    6.800  179.0
16             Canguro   35.000   56.0
17             Criceto    0.120    1.0
18                Topo    0.023    0.4
19            Coniglio    2.500   12.1
20              Pecora   55.500  175.0
21            Giaguaro  100.000  157.0
22           Scimpanze   52.160  440.0
23               Ratto    0.280    1.9
24               Talpa    0.122    3.0
25              Maiale  192.000  180.0
```

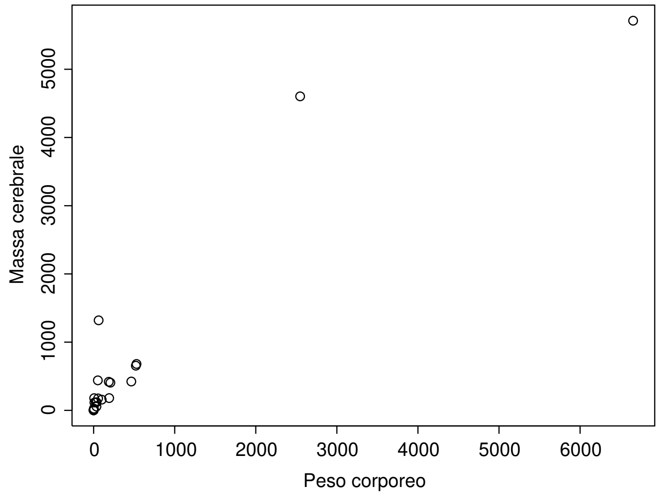

Figura 8.15: Grafico del peso corporeo e della massa cerebrale.

Costruiamo due vettori, il primo contenente il peso corporeo e il secondo la massa cerebrale:

```
> peso<-animali[,2]
> massa<-animali[,3]
>
```

Osserviamo nel grafico che segue quale possibile modello potrebbe descrivere la relazione:

```
> plot(peso,massa,xlab="Peso corporeo",ylab="Massa cerebrale")
>
```

Il grafico è riportato in figura 8.15. Si vede che una regressione lineare del tipo:

$$massa = \alpha + \beta peso$$

potrebbe essere adeguata. Passiamo allora al calcolo di α e β attraverso il metodo dei minimi quadrati.

```
> modello.lin<-lm(massa~peso)
> summary(modello.lin)
```

```
>
Call:
lm(formula = massa ~ peso)

Residuals:
    Min      1Q  Median      3Q     Max
-755.05 -188.34 -128.54  -18.64 2009.53

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 191.22788  110.08767   1.737   0.0958   .
peso          0.94316    0.07658  12.317 1.31e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 520.5 on 23 degrees of freedom
Multiple R-Squared: 0.8683,     Adjusted R-squared: 0.8626
F-statistic: 151.7 on 1 and 23 degrees of freedom, p-value: 1.312e-11

>
```

Aggiungiamo al primo grafico la retta di regressione con il comando:

```
> parametri<-coefficients(modello.lin)
>
> plot(peso,massa,xlab="Peso corporeo",ylab="Massa cerebrale",
+ xlim=c(0,600),ylim=c(0,1500))
>
> abline(parametri[1],parametri[2])
>
```

L'angolo in basso a destra di questo grafico è riportato in figura 8.16. Come si può notare il valore di β è positivo e indica che un corpo più pesante presenta una massa cerebrale più elevata. L'intercetta così elevata tuttavia mostra come i piccoli animali siano mal rappresentati, perché, pesi corporei al di sotto del kilogrammo secondo il modello hanno una massa cerebrale intorno ai 200 grammi. Si veda ad esempio il maiale della guina che pesa circa un kilogrammo ed ha una massa cerebrale pari 5.5 grammi, secondo il modello ha una massa cerebrale pari a 192 grammi. Questo fatto è possibile notarlo anche dal grafico dei residui:

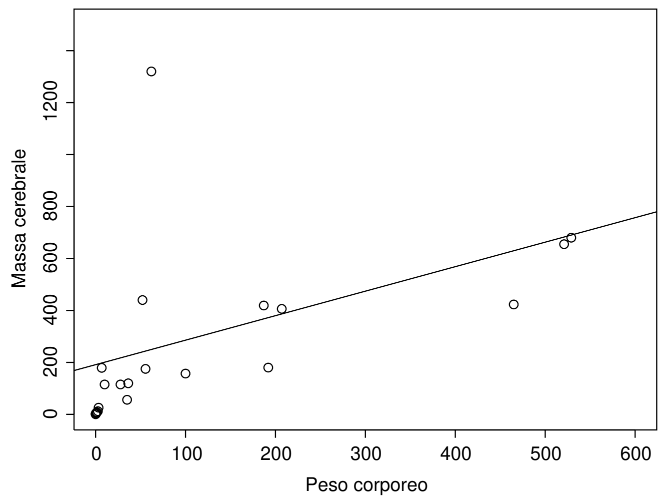

Figura 8.16: Grafico del modello insieme ai dati.

```
> massa.prevista<-fitted.values(modello.lin)
> residui<-residuals(modello.lin)
> residui[1:10]
           [,1]
 [1,] -184.40115
 [2,] -206.79956
 [3,] -105.96477
 [4,] -102.31582
 [5,] -186.70877
 [6,] 2009.53110
 [7,]   51.40028
 [8,]  -27.61680
 [9,]  -85.65953
[10,] -168.74033
>
```

Un loro grafico è in figura 8.17.

```
> plot(massa.prevista,residui,xlab="Massa cerebrale prevista",
+ ylab="Residui")
```

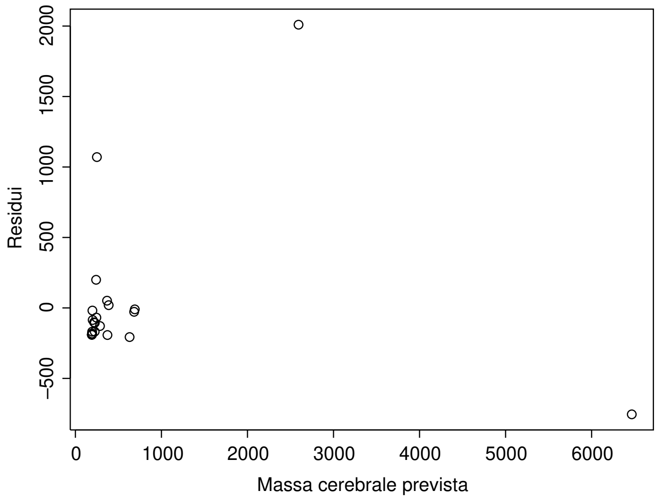

Figura 8.17: Valori previsti e residui del modello.

```
>
```

Il grafico dei residui contro i valori previsti, evidenzia la presenza di alcuni residui elevati, il primo in corrispondenza dell'elefante asiatico, il secondo riferito all'uomo dove, il valore previsto è pari a 250 grammi contro i 1320 osservati. Il grafico non è del tutto soddisfacente.

Questo è dovuto principalmente al fatto che il fenomeno analizzato varia su diversi ordini di grandezza. Per ovviare a questo problema, alla presenza di punti leva e per migliorare il grafico dei residui può essere interessante analizzare una trasformazione di entrambe le variabili su scala logaritmica in base 10. Calcoliamo le nuove variabili:

```
> logpeso<-log10(peso)
> logmassa<-log10(massa)
```

e rappresentiamole graficamente:

```
> plot(logpeso,logmassa,xlab="Logaritmo del peso corporeo",
+ ylab="Logaritmo della massa cerebrale")
>
```

Il grafico è riportato in figura 8.18. In questo grafico risulta evidente la possibilità di usare una retta di regressione:

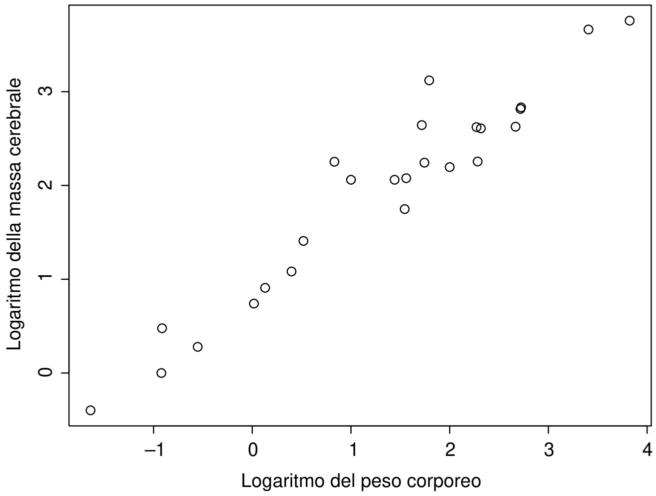

Figura 8.18: Grafico dei dati trasformati.

```
> modello.log<-lm(logmassa~logpeso)
> summary(modello.log)
>
Call:
lm(formula = logmassa ~ logpeso)

Residuals:
     Min       1Q   Median       3Q      Max
-0.39630 -0.20637 -0.06762  0.08426  0.83830

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.93392    0.08712   10.72 2.03e-10 ***
logpeso      0.75227    0.04572   16.45 3.24e-14 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.3152 on 23 degrees of freedom
Multiple R-Squared: 0.9217,     Adjusted R-squared: 0.9183
F-statistic: 270.7 on 1 and 23 degrees of freedom, p-value: 3.242e-14
```

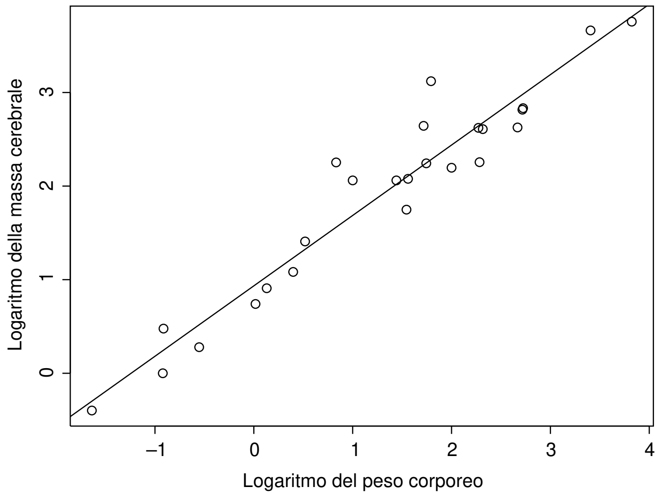

Figura 8.19: Modello ottenuto dai dati trasformati.

```
>
> logpar<-coefficients(modello.log)
>
```

aggiungiamo al grafico precedente la retta di regressione:

```
> plot(logpeso,logmassa,xlab="Logaritmo del peso corporeo",
+ ylab="Logaritmo della massa cerebrale")
>
> abline(logpar[1],logpar[2])
>
```

Il grafico è riportato in figura 8.19. Calcoliamo i valori previsti e i residui del nuovo modello:

```
> logmassa.prevista<-fitted.values(modello.log)
> logres<-residuals(modello.log)
>
```

Il grafico (riportato in figura 8.20) che rappresenta i valori previsti contro i residui evidenzia la mancanza di un qualche andamento non casuale:

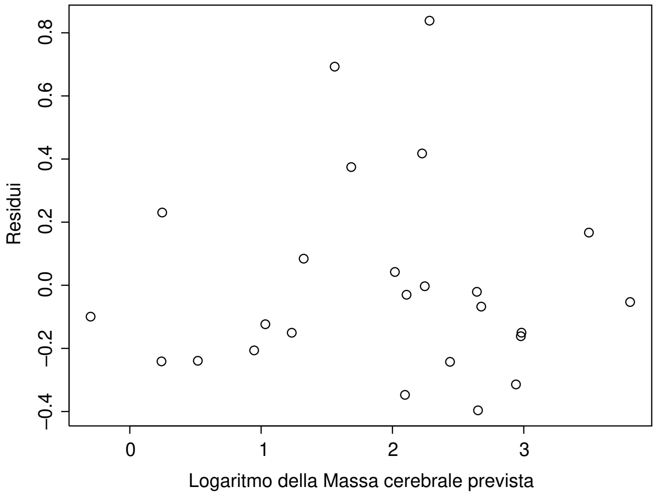

Figura 8.20: Valori previsti e residui sui dati trasformati.

```
> plot(logmassa.prevista,logres,
+ xlab="Logaritmo della Massa cerebrale prevista",
+ ylab="Residui")
>
```

Si noti che la retta di regressione che abbiamo appena studiato è pari a:

$$\log_{10} massa = \gamma + \delta \log_{10} peso$$

e corrisponde alla seguente relazione non lineare sulle variabili originali:

$$massa = 10^{\gamma} \times peso^{\delta}$$

da cui è immediato notare che a un peso corporeo nullo corrisponde una massa cerebrale nulla. Con questa formula possiamo calcolare i valori previsti dal modello nelle variabili originali:

```
> massa.prevista.10<-10^logpar[1] * peso^logpar[2]
> massa.prevista.10
[1]  10.763875 872.081404 128.052375 104.370868 8.845790 3134.459006
[7] 439.493121 949.964786 48.549968 21.085475 960.917194 474.404313
[13] 191.548271 6455.026137 36.323786 124.587024 1.742699 0.502929
```

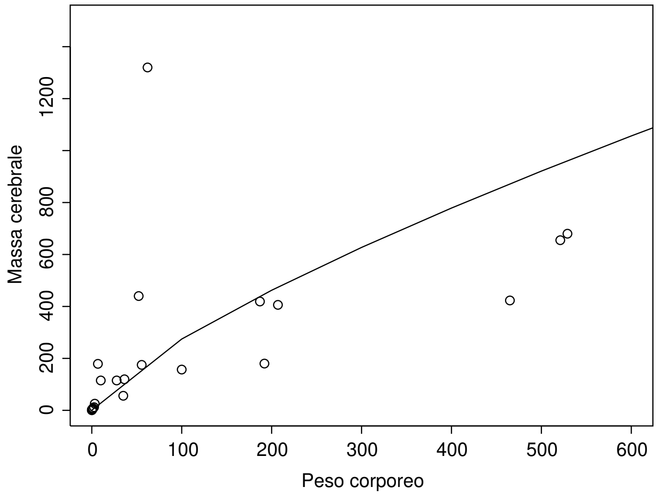

Figura 8.21: Modello calcolato sui dati trasformati riportato sui dati originali.

```
[19] 17.111171 176.236239 274.444731 168.196745 3.296392 1.764504
[25] 448.304156
>
```

da cui si nota come siano migliarati i valoti previsti per gli animali con un peso notevole e sicuramente quelle con un peso corporeo molto piccolo, in particolare il valore previsto per il maiale della guinea è pari a 8.84 grammi contro i 192.2 della retta di regressione. Il grafico di questa regressione (figura 8.21) si ottiene come segue:

```
> linea<-seq(min(peso),max(peso),by=100)
> plot(peso,massa,xlab="Peso corporeo",
+ ylab="Massa cerebrale",
+ xlim=c(0,600),ylim=c(0,1500))
>
> lines(linea,10^logpar[1] * linea^logpar[2])
>
```

Si noti infine che la somma dei residui dalla seconda regressione espressi nelle variabili originarie non è zero:

```
> residui.10<-massa-massa.prevista.10
```

```
> sum(residui.10)
[1] 760.027
>
```

8.5 Hald Cement Data

Questo insieme di dati è composto delle seguenti variabili:

X_1 *tricalcium aluminate*

X_2 *tricalcium silicate*

X_3 *tetracalcium alumino ferrite*

X_4 *dicalcium silicate*

Y *heat evolved*, Calore sviluppato (calorie per grammi).

Si vuole capire se è possibile valutare il calore sviluppato in base agli ingredienti utilizzati per la composizione del cemento. È noto che alcuni di questi elementi sono tra loro in parte equivalenti.

Cominciamo leggendo le osservazioni che si trovano nel *file* `"hald.txt"` e costruiamo un vettore contenente la variabile dipendente e una matrice contenente le variabili esplicative:

```
> hald<-read.table("hald.txt")
>
> calore<-hald[,1]
> componenti<-hald[,2:5]
>
> dimnames(componenti)<-list(1:nrow(componenti),c("X1","X2","X3","X4"))
>
```

La funzione `dimnames()` attribuisce il nome alle righe e alle colonne della matrice `componenti`.

Osserviamo attraverso un grafico (figura 8.22) quale possa essere la relazione tra la variabile dipendente e ognuna delle possibili variabili esplicative:

```
> par(mfcol=c(2,2))
>
```

```
> plot(componenti[,1],calore,xlab="tricalcium aluminate",
+ ylab="calore sviluppato")
> plot(componenti[,2],calore,xlab="tricalcium silicate",
+ ylab="calore sviluppato")
> plot(componenti[,3],calore,xlab="tetracalcium alumino ferrite",
+ ylab="calore sviluppato")
> plot(componenti[,4],calore,xlab="dicalcium silicate",
+ ylab="calore sviluppato")
>
```

La funzione `par()` consente di modificare i parametri riguardanti la grafica. Abbiamo modificato il parametro `mfcol` che serve per suddividere la finestra grafica in diverse porzioni. In questo caso otteniamo una finestra costituita da una matrice con due righe e due colonne. Dalla figura vediamo che le relazioni si possono considerare lineari. Diventa importante capire quali possono essere le variabili rilevanti.

A questo scopo analizziamo la matrice di correlazione delle variabili esplicative:

```
> cor(componenti)
           X1         X2          X3          X4
X1  1.0000000  0.2285795 -0.82413376 -0.24544511
X2  0.2285795  1.0000000 -0.13924238 -0.97295500
X3 -0.8241338 -0.1392424  1.00000000  0.02953700
X4 -0.2454451 -0.9729550  0.02953700  1.00000000
>
```

e la correlazione che vi è tra la variabile dipendente e le variabili esplicative:

```
> cor(calore,componenti)
            X1        X2         X3        X4
[1,] 0.7307175 0.8162526 -0.5346707 -0.821305
>
```

Si noti come la prima variabile è fortemente correlata con la terza mentre la seconda variabile è fortemente correlata con la quarta. La variabile candidata è l'ultima variabile visto il suo elevato valore di correlazione con la variabile dipendente.

Stimiamo il modello:

$$calore = a + b\ X_4 + \varepsilon$$

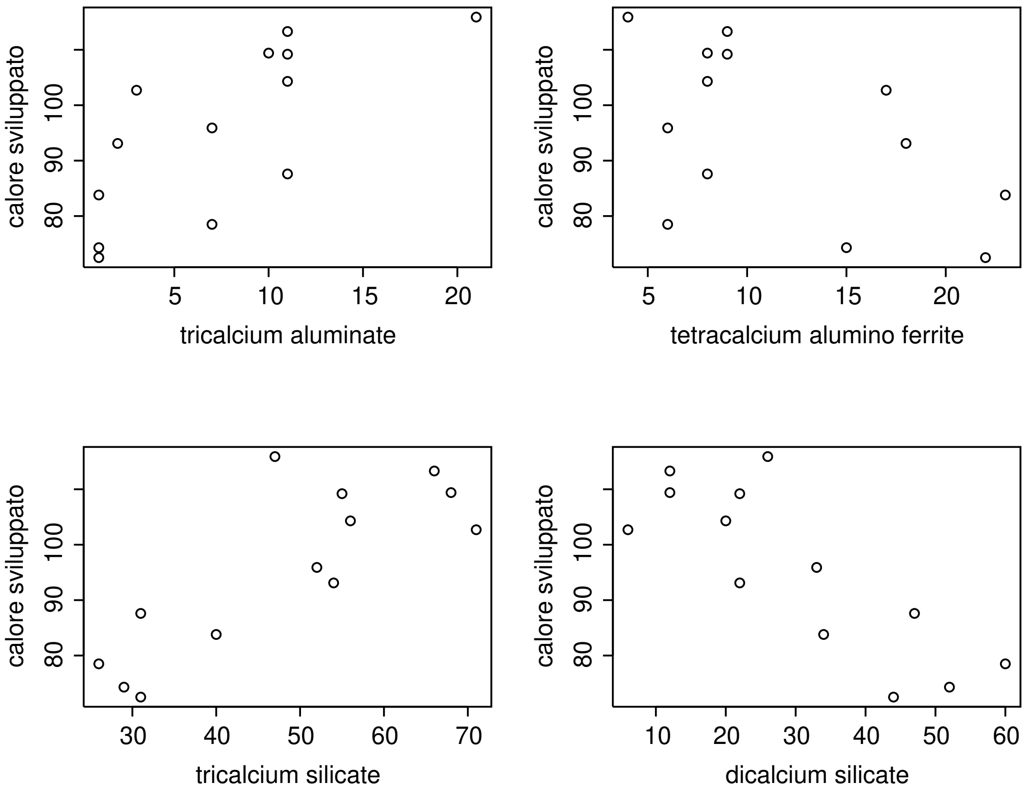

Figura 8.22: Grafici tra la variabile dipendente e le possibili variabili esplicative.

attraverso il metodo dei minimi quadrati

```
> modello.uno<-lm(calore~componenti[,4])
summary(modello.uno)>

Call:
lm(formula = calore ~ componenti[, 4])

Residuals:
    Min      1Q  Median      3Q     Max
-12.589  -8.228   1.495   4.726  17.524

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     117.5679     5.2622  22.342 1.62e-10 ***
componenti[, 4]  -0.7382     0.1546  -4.775 0.000576 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 8.964 on 11 degrees of freedom
Multiple R-Squared: 0.6745,     Adjusted R-squared: 0.645
F-statistic:  22.8 on 1 and 11 degrees of freedom, p-value: 0.0005762
```

Definiamo i valori previsti e i residui di questo modello:

```
> pre.modello.uno<-fitted.values(modello.uno)
> res.modello.uno<-residuals(modello.uno)
>
```

Nella figura 8.23 riportiamo il grafico dei valori previsti contro i residui del primo modello, notiamo che i residui non presentano un particolare andamento. Prima di produrre il grafico, ripristiniamo la configurazione della finestra.

```
> par(mfcol=c(1,1))
>
> plot(pre.modello.uno,res.modello.uno,
+ xlab="Valori previsti con il primo modello",
+ ylab="Residui del primo modello")
>
```

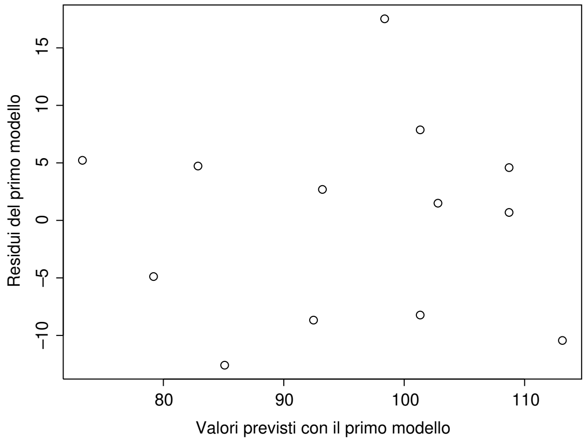

Figura 8.23: Grafico dei valori previsti contro i residui del primo modello.

Calcoliamo ora la correlazione tra i residui e le variabili esplicative al netto della variabile che è nel modello. Questo è chiamato coefficiente di correlazione parziale e misura la dipendenza lineare che vi è tra due variabili al netto di una terza. Se ad esempio vogliamo calcolare la correlazione parziale tra z_1 e z_2 al netto di z_3 è sufficiente calcolare la correlazione tra i residui della regressione di z_1 su z_3 e i residui della regressione di z_2 su z_3 quindi:

```
> temp.uno<-residuals(lm(componenti[,1]~componenti[,4]))
> temp.due<-residuals(lm(componenti[,2]~componenti[,4]))
> temp.tre<-residuals(lm(componenti[,3]~componenti[,4]))
>
> cor(res.modello.uno,cbind(temp.uno,temp.due,temp.tre))
         [,1]      [,2]       [,3]
[1,] 0.956773 0.1302149 -0.8950818
>
```

La variabile X_1 ha un coefficiente di correlazione parziale molto elevato e quindi è la variabile che riesce a descrivere parte della variabilità della variabile dipendente che X_4 non riesce a cogliere. Proviamo a stimare il modello inserendo la prima variabile

```
> modello.due<-lm(calore~componenti[,4]+componenti[,1])
> summary(modello.due)
```

```
Call:
lm(formula = calore ~ componenti[, 4] + componenti[, 1])

Residuals:
    Min      1Q  Median      3Q     Max
-5.0234 -1.4737  0.1371  1.7305  3.7701

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     103.09738    2.12398   48.54 3.32e-13 ***
componenti[, 4]  -0.61395    0.04864  -12.62 1.81e-07 ***
componenti[, 1]   1.43996    0.13842   10.40 1.11e-06 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 2.734 on 10 degrees of freedom
Multiple R-Squared: 0.9725,     Adjusted R-squared: 0.967
F-statistic: 176.6 on 2 and 10 degrees of freedom, p-value: 1.581e-08
```

Si vede come il coefficiente di determinazione R^2 sia aumentato notevolmente e come i parametri del nuovo modello risultino tutti diversi da zero.

Proviamo a vedere se una nuova variabile può portare un ulteriore contributo al miglioramento del modello. A questo scopo calcoliamo il coefficiente di correlazione parziale tra la variabile dipendente e X_2 e X_3 al netto di X_1 e X_4 quindi:

```
> temp.due<-residuals(lm(componenti[,2]~componenti[,1]
+ +componenti[,4]))
> temp.tre<-residuals(lm(componenti[,3]~componenti[,1]
+ +componenti[,4]))
>
> cor(res.modello.due,cbind(temp.due,temp.tre))
          [,1]       [,2]
[1,] 0.5986053 -0.5657105
>
```

Da cui notiamo che le correlazioni sono ancora abbastanza elevate, proviamo quindi ad inserire nel modello la variabile X_2:

```
> modello.tre<-lm(calore~componenti[,4]+componenti[,1]
+ +componenti[,2])
> summary(modello.tre)

Call:
lm(formula = calore ~ componenti[, 4] + componenti[, 1] + componenti[,
    2])

Residuals:
    Min      1Q  Median      3Q     Max
-3.0919 -1.8016  0.2562  1.2818  3.8982

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)      71.6483    14.1424   5.066 0.000675 ***
componenti[, 4]  -0.2365     0.1733  -1.365 0.205395
componenti[, 1]   1.4519     0.1170  12.410 5.78e-07 ***
componenti[, 2]   0.4161     0.1856   2.242 0.051687   .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.309 on 9 degrees of freedom
Multiple R-Squared: 0.9823,     Adjusted R-squared: 0.9764
F-statistic: 166.8 on 3 and 9 degrees of freedom, p-value: 3.323e-08
```

Dal test t notiamo che il coefficiente della variabile X_4 può essere considerato nullo quindi stimiamo di nuovo il modello togliendo X_4:

```
> modello.quattro<-lm(calore~componenti[,1]+componenti[,2])
> summary(modello.quattro)

Call:
lm(formula = calore ~ componenti[, 1] + componenti[, 2])

Residuals:
   Min     1Q Median     3Q    Max
-2.893 -1.574 -1.302  1.362  4.048

Coefficients:
```

```
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     52.57735    2.28617   23.00 5.46e-10 ***
componenti[, 1]  1.46831    0.12130   12.11 2.69e-07 ***
componenti[, 2]  0.66225    0.04585   14.44 5.03e-08 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 2.406 on 10 degrees of freedom
Multiple R-Squared: 0.9787,     Adjusted R-squared: 0.9744
F-statistic: 229.5 on 2 and 10 degrees of freedom, p-value: 4.407e-09
```

Notate come R^2 sia sceso solo di poco. Calcoliamo i coefficienti di correlazione parziale:

```
> temp.tre<-residuals(lm(componenti[,3]~componenti[,1]
+ +componenti[,2]))
> temp.quattro<-residuals(lm(componenti[,4]~componenti[,1]
+ +componenti[,2]))
>
> cor(res.modello.quattro,cbind(temp.tre,temp.quattro))
          [,1]       [,2]
[1,] 0.4112643 -0.4141492
>
```

Inseriamo infine la terza variabile nel modello:

```
> summary(lm(calore~componenti[,1]+componenti[,2]+componenti[,3]))

Call:
lm(formula = calore ~ componenti[, 1] + componenti[, 2] + componenti[,
    3])

Residuals:
    Min      1Q  Median      3Q     Max
-3.2543 -1.4726  0.1755  1.5409  3.9711

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)     48.19363    3.91330  12.315 6.17e-07 ***
```

```
componenti[, 1]  1.69589    0.20458   8.290 1.66e-05 ***
componenti[, 2]  0.65691    0.04423  14.851 1.23e-07 ***
componenti[, 3]  0.25002    0.18471   1.354    0.209
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.312 on 9 degrees of freedom
Multiple R-Squared: 0.9823,     Adjusted R-squared: 0.9764
F-statistic: 166.3 on 3 and 9 degrees of freedom, p-value: 3.367e-08
```

Ma il suo coefficiente può essere considerato nullo, e sappiamo che la quarta variabile è già stata esclusa, quindi il modello migliore è:

$$calore = a + b\ X_1 + c\ X_2 + \varepsilon$$

Il grafico dei residui è (figura 8.24):

```
> plot(pre.modello.quattro,res.modello.quattro,
+ xlab="Valori previsti con il quarto modello",
+ ylab="Residui del terzo modello")
>
```

da cui notiamo che non si presenta nessun particolare andamento. Il modello che abbiamo individuato può essere considerato un buon modello, tuttavia, le assunzioni che abbiamo fatto sulla struttura del termine di errore non sembrano verificate. Infatti, supponiamo che la distribuzione sia simmetrica, mentre da un diagramma a scatola e baffi e da un istrogramma (figura 8.25) vediamo che la distribuzione sembra asimmetrica:

```
> par(mfcol=c(1,2))
>
> boxplot(res.modello.quattro,xlab="residui",
+ main="Diagramma a scatola e baffi")
>
> hist(res.modello.quattro,xlab="residui",main="Istogramma")
>
```

Inoltre il termine d'errore non è ben approssimato da un modello teorico normale come si vede dal grafico in figura 8.26 costruito confrontando i quantili dei residui con i quantili di una normale standard:

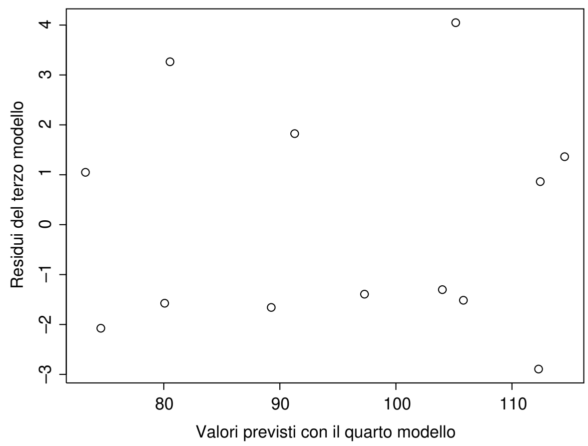

Figura 8.24: Grafico dei valori previsti contro i residui del quarto modello.

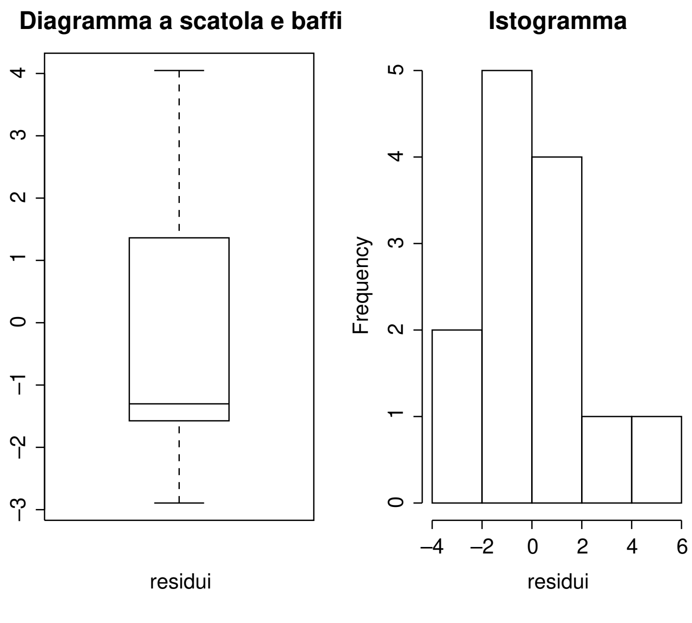

Figura 8.25: Diagramma a scatola e baffi e istogramma dei residui del quarto modello.

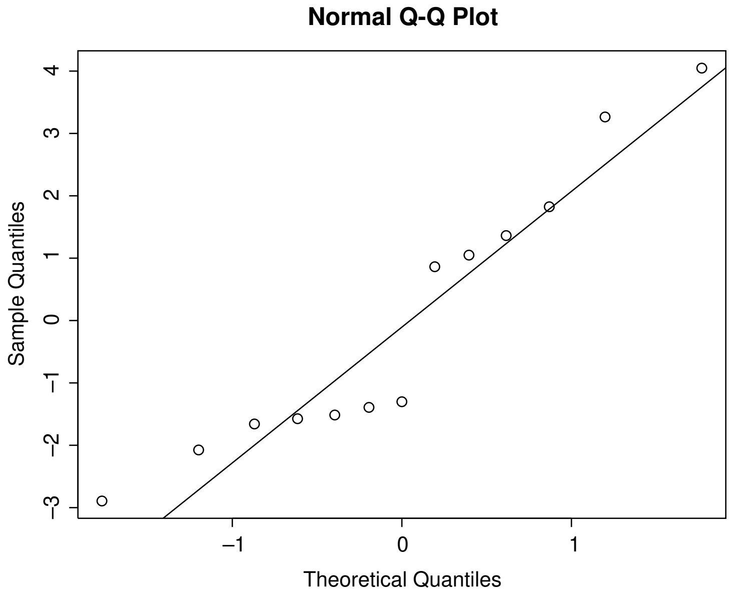

Figura 8.26: Grafico quantile-quantile per i residui del quarto modello.

```
> par(mfcol=c(1,1))
>
> qqnorm(res.modello.quattro)
> qqline(res.modello.quattro)
```

8.6 Spedizioni

Nel *file* "spedizioni.txt" sono state raccolte da una ditta i costi di spedizione in lire per ettogrammo di merce trasportata e la distanza alla quale è stata spedita. Il costo di spedizione pùo variare a parità di distanza, dalla nazione in cui la merce viene spedita, sia per motivi doganali, sia perchè il traffico con un determinato paese può risultare particolarmente disagievole.

Leggiamo questo insieme di dati e costruiamo i due vettori contenenti rispettivamente la distanza e il costo di spedizione:

```
> spedizioni<-read.table(file="spedizioni.txt")
>
```

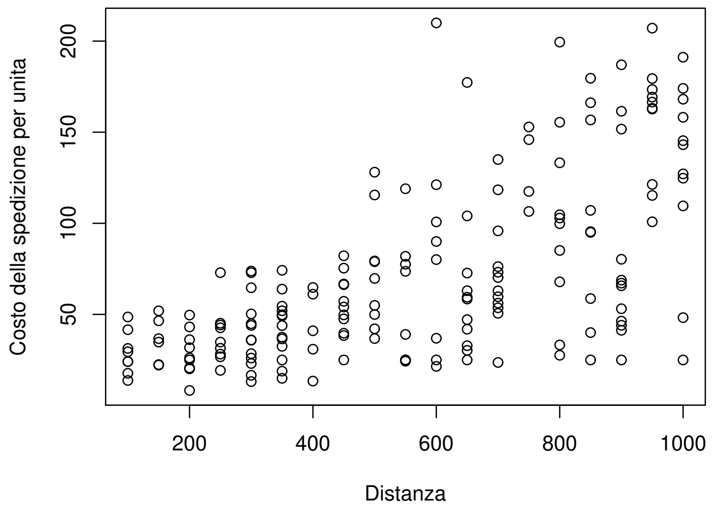

Figura 8.27: Grafico del costo di spedizione e della distanza.

```
> distanza<-spedizioni[,1]
> costo<-spedizioni[,2]
>
```

Attraverso il grafico di figura 8.27 notiamo che vi potrebbe essere una relazione lineare, al più una relazione quadratica.

```
> plot(distanza,costo,xlab="Distanza",
+ ylab="Costo della spedizione per unita")
>
```

Stimiamo con il metodo dei minimi quadrati il modello quadratico del tipo:

$$costo = a + b\ distanza + c\ distanza^2 + \varepsilon$$

```
> distanza2<-distanza^2
>
> modello.1<-lm(costo~distanza+distanza2)
> summary(modello.1)
```

```
Call:
lm(formula = costo ~ distanza + distanza2)

Residuals:
     Min       1Q   Median       3Q      Max
-106.898  -19.827   -3.525   19.394  140.120

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.607e+01  1.230e+01   2.119   0.0355 *
distanza    2.373e-02  4.962e-02   0.478   0.6331
distanza2   8.206e-05  4.302e-05   1.908   0.0581 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 37.6 on 175 degrees of freedom
Multiple R-Squared: 0.4286,     Adjusted R-squared: 0.4221
F-statistic: 65.63 on 2 and 175 degrees of freedom, p-value:  0
```

dove notiamo che l'unico coefficiente che può essere considerato sicuramente non nullo è quello relativo all'intercetta. Analizziamo i residui di questa regressione:

```
> res.1<-residuals(modello.1)
>
> plot(distanza,res.1,xlab="Distanza",ylab="Residui")
>
```

Dal grafico in figura 8.28 possiamo notare come l'ipotesi di omoschedasticità fatta sul termine di errore sia palesemente non verificata. Infatti al crescere della distanza aumenta la variabilità dei residui. Per migliorare la precisione delle stime del nostro modello è bene tener conto di questo fatto. Andiamo quindi a stimare la varianza della variabile `costo` rispetto al valore assunto dalla variabile `distanza`, calcoliamo cioè la varianza condizionata.

Costruiamo dapprima un vettore contenente i valori distinti della variabile `distanza` e calcoliamo la frequenza di osservazioni della variabile `costo` che si riferisce allo stessa `distanza`:

```
> distinti<-seq(100,1000,50)
```

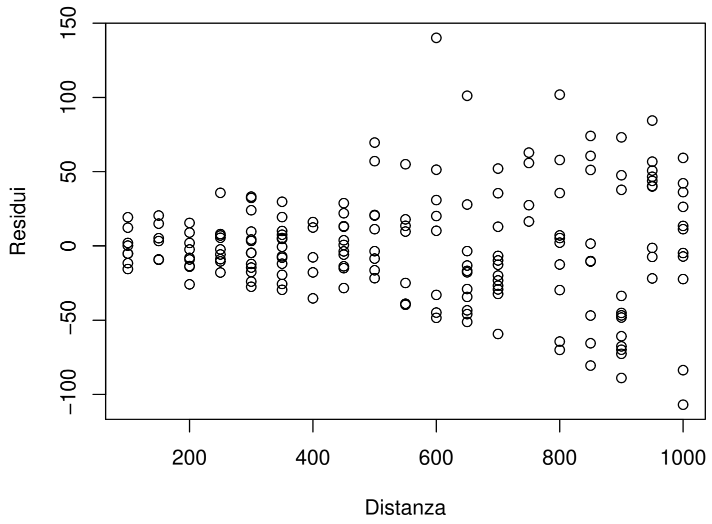

Figura 8.28: Grafico dei residui - distanza.

```
> freq<-table(distanza)
>
```

Possiamo calcolare le varianze condizionate con il seguente algoritmo che sfrutta un ciclo, definito dal comando `for()`:

```
> varianze<-0
> for(i in 1:19){
+ dati<-costo[distanza==distinti[i]]
+ n<-length(dati)
+ varianze<-c(varianze,var(dati)*(n-1)/n)
+ }
> varianze<-varianze[-1]
>
```

Il grafico di figura 8.29 ci conferma che le varianze condizionate non sono uguali, queste tendono a crescere al crescere della distanza:

```
> plot(distinti,varianze,xlab="Distanze",ylab="Varianza condizionata")
>
```

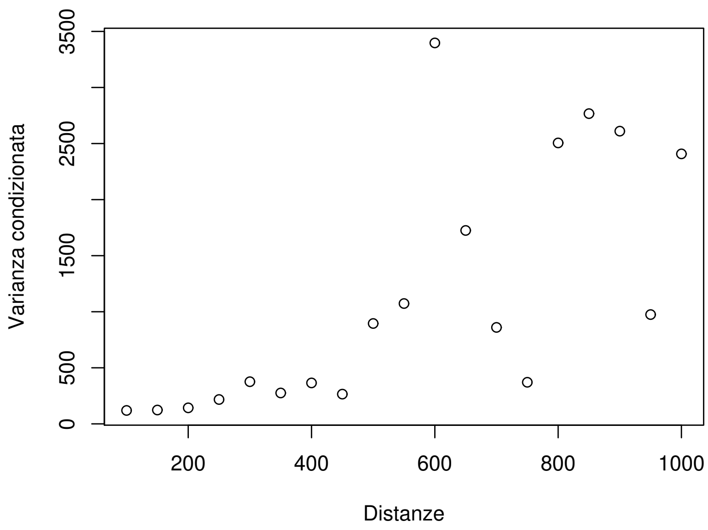

Figura 8.29: Grafico delle varianze condizionate - distanza.

Costruiamo un vettore che contenga la corrispondente varianza condizionata per ogni osservazione e definiamo il vettore dell'errore quadratico medio. Successivamente possiamo ristimare il modello utilizzando i minimi quadrati pesati.

```
>varianze.vett_rep(varianze,times=freq)
>scarto_sqrt(varianze.vett)

> modello.2_lm(costo~distanza+distanza2, weights=(1/scarto))
> summary(modello.2)

Call:
lm(formula = costo ~ distanza + distanza2, weights = (1/scarto))

Residuals:
     Min       1Q   Median       3Q      Max
-16.1758  -4.3532  -0.8187   3.7785  18.2359

Coefficients:
```

```
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 2.651e+01  7.828e+00   3.387 0.000874 ***
distanza    1.668e-02  3.633e-02   0.459 0.646756
distanza2   9.510e-05  3.400e-05   2.797 0.005733 **
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.134 on 175 degrees of freedom
Multiple R-Squared: 0.5326,     Adjusted R-squared: 0.5273
F-statistic: 99.72 on 2 and 175 degrees of freedom, p-value: 0
```

Il risultato è più soddisfacente, si noti come avendo eliminato l'eteroschedasticità si sia migliorata la precisione delle stime e risulti più chiara quale potrebbe essere la relazione esistente tra le due variabili. Un modello finale potrebbe escludere la variabile distanza.