Matematica e Statistica con R

Federico Comoglio e Maurizio Rinaldi

29 gennaio 2016

Indice

Capitolo 1

Statistica con R

1.1 Variabili aleatorie

Una variabile aleatoria (*random variabile*) è una variabile i cui valori sono soggetti a variazioni casuali. Quando i valori possibili di una variabile aleatoria possono essere elencati parliamo di variabile aleatoria discreta. Quando i valori non possono essere elencati parliamo di variabile aleatoria continua.

1.2 Variabili aleatorie discrete

Le variabili aleatorie discrete che assumono un numero limitato di valori si dicono anche *finite*. I valori di una variabile aleatoria discreta possono essere numerici o nominali. Supponiamo di avere una variabile aleatoria che possa assumere un insieme di valori in un *alfabeto* assegnato costituito da lettere, parole o numeri. Per esempio un alfabeto può essere del tipo che segue

- (Femmina, Maschio)
- (A,C,T,G)
- (0,1)
- (Ottimo, Buono, Discreto, Sufficiente, Insufficiente)
- (Testa, Croce).
- I numeri interi

Per caratterizzare completamente una variabile aleatoria discreta oltre ai valori che questa può assumere occorre conoscere la probabilità di questi valori.

Per semplicità considereremo variabili aleatorie finite.
Come possiamo simulare variabili aventi valore nell'alfabeto assegnato? In effetti qualunque

comando di generazione su un computer non è perfettamente casuale; infatti la generazione avviene in effetti in modo pseudo-casuale e secondo un meccanismo che dipende dallo stato interno del computer codificato in una variabile indicata con `.Random.seed`. Se il *seme* iniziale è lo stesso i numeri generati saranno uguali. Spesso conviene che i calcoli (ad esempio a fine didattico) siano riproducibili. Ad esempio mettendo in una variabile `seme` il valore corrente di `.Random.seed` e richiamandolo o generandolo all'occorrenza. Scegliamo per la riproducibilità dei risultati

```
> seme=as.integer(c(0,1,2,3))
```

A questo punto possiamo simulare le variabili richieste usando la struttura

$$\texttt{sample}(\textit{alfabeto}, n) \tag{1.1}$$

Se l'alfabeto consiste di tutte le lettere minuscole dell'alfabeto ordinario e ne vogliamo selezionare $n = 8$ (in modo che ciascun uscita abbia la stessa probabilità) basta scrivere

```
> sample(letters,8)
[1] "o" "r" "j" "g" "z" "x" "m" "u"
```

Se invece l'alfabeto consiste delle basi del DNA

```
> alfabeto=c("A","C","G","T")
> sample(alfabeto,2)
[1] "C" "G"
>
```

Notiamo che

```
>  sample(alfabeto)
[1] "C" "G" "T" "A"
```

restituisce una permutazione dell'alfabeto, mentre chiedendo un campione di lunghezza superiore alla lunghezza dell'alfabeto otteniamo un messaggio di errore. Possiamo però immaginare di re-immettere la lettera estratta nell'urna dopo ogni estrazione. In questo caso non c'è limite alla sequenza generata. Per esempio

```
> alfabeto=c("testa","croce")
> sample(alfabeto,5,replace=T)
[1] "testa" "testa" "testa" "croce" "croce"
```

Il precursore del dado era chiamato astragalo ed era giocato nell'antica Grecia e nell'antica Roma [**?**]. Gli astragali sono dei piccoli ossicini di forma irregolare ed hanno 6 facce ma atterranno in modo stabile solo su 4 di esse numerate 1, 3, 4 e 6 con probabilità all'incirca 0.4 per il 3 e il 4 e di 0.1 per l'1 e il 6. Il tiro più gettonato all'epoca era l'uscita di 4 facce diverse nel lancio di 4 astragali e si chiamava *Venus*. Il lancio considerato peggiore sul singolo lancio era l'1 chiamato cane o avvoltoio. Per simulare un astragalo su un computer

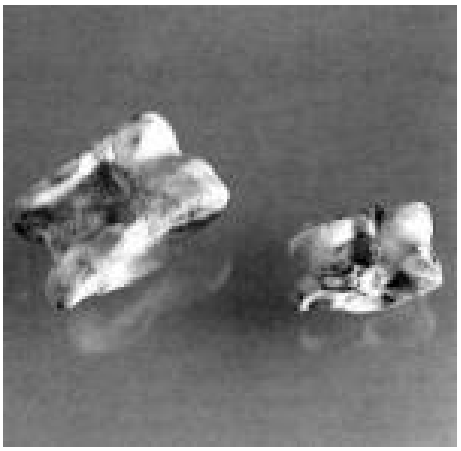

Figura 1.1: Astragalo.

```
> sample(c(1,3,4,6),4,replace=T,prob=c(0.1,0.4,0.4,0.1))
[1] 4 4 4 4
```

Torniamo ora ai classici dadi a 6 facce. Supponiamo di lanciare 100 volte un dado equo a 6 facce e di registrare in `x` le uscite rilevate

```
> .Random.seed=seme
> dadi100<-sample(1:6,100,replace=T)
> dadi100
  [1] 1 5 1 5 3 6 5 5 5 6 6 3 5 6 6 5 5 2 4 6 1 4 2 3 4 5 2 4
 [29] 5 6 3 3 5 5 6 1 6 4 2 6 6 3 2 2 3 5 1 3 3 2 5 6 3 4 2 4
 [57] 2 1 4 1 6 5 6 2 2 3 4 3 3 4 3 2 4 3 6 3 1 3 2 4 6 1 3 1
 [85] 1 6 6 6 5 5 3 3 3 4 1 4 4 2 5 3
```

Volendo invece simulare una combinazione da giocare al SuperEnalotto possiamo scrivere

```
> x<-sample(1:90,6,replace=T);x
[1] 13 76 66 76 17 53
```

I numeri usciti sono stati salvati in una variabile `x`, per poter effettuare la ricerca di indicatori statistici. Il comando che consente di ordinare una lista o un vettore è `sort`, esso può essere usato in associazione al nome di una variabile o di una lista, ossia:

$$\texttt{sort}(\textit{variabile/lista}) \tag{1.2}$$

Volendo ordinare i numeri precedentemente ricavati scriveremo

```
> sort(x)
[1] 13 17 53 66 76 76
```

1.3 Statistica descrittiva: singola variabile

1.3.1 Indicatori statistici

- Media.

  La media di una serie di numeri si ottiene con la funzione `mean` scrivendo: `mean`(*variabile*). Ad esempio, lavorando con la lunghezza del sepalo di 150 piante di iris

```
> mean(x=iris[,1])
[1] 5.843333
```

- Varianza campionaria

  Si ottiene con la funzione predefinita di espressione: `var`(*variabile*). Possiamo calcolare la varianza come

```
> var(x)
[1] 814.9667
```

- Deviazione Standard campionaria.

  Non è altro che la radice della varianza. Si ottiene con la funzione predefinita di espressione: `sd`(*variabile*). Sempre basandosi sull'esempio precedente scriveremo

```
> sd(x)
[1] 28.54762
```

- Quantili. La notazione standard è semplicemente: `quantile`(*variabile*) che determina i quartili e ci fornisce in uscita la statistica dei 5 numeri

```
> quantile(x)
  0%  25%  50%  75% 100%
13.0 26.0 59.5 73.5 76.0
```

  Volendo ricavare i decili dovremo scrivere:

$$\texttt{quantile}(\textit{variabile}, \texttt{seq(0,1,by=0.1)})$$

  in quanto vogliamo dividere l'intervallo $[0, 1]$ a passo 0.1

  Nell'esempio:

```
> quantile(x,seq(0,1,by=0.1))

  0%  10%  20%  30%  40%  50%  60%  70%  80%  90% 100%
13.0 15.0 17.0 35.0 53.0 59.5 66.0 71.0 76.0 76.0 76.0
```

Si noti che `quantile` ammette 9 varianti specificabili con l'opzione `type` $= n$ dove n va da 1 a 9. Per esempio

```
> quantile(x,type=4)

  0%  25%  50%  75% 100%
  13   15   53   71   76
```

Sui dati in esame le 9 varianti coincidono.

Per quanto riguarda gli indicatori statistici nel caso di dati ripetuti basta notare che se la lista x contiene i valori e la lista f le frequenze assolute il comando

$$\mathit{rep}(x, f)$$

costruisce un'unica lista dei dati inclusiva delle ripetizioni. Per esempio

```
> x=1:6
> f=c(9,7,9,7,8,10)
> y=rep(x,f)
> y
 [1] 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 4 4 4 4 4
[31] 4 4 5 5 5 5 5 5 5 5 6 6 6 6 6 6 6 6 6 6
```

Ovviamente senza bisogno di visualizzare `y` possiamo calcolarne tutti gli indicatori statistici. Il comando

```
> cumsum(f)
[1]  9 16 25 32 40 50
```

restituisce le frequenze cumulate, dalle quali si possono ricavare facilmente la mediana i quantili.

1.3.2 Raggruppamenti in classi

```
> mese=month(as.POSIXlt(date(),format="%a %b %d %H:%M:%S %Y"))
> mesit= c("Dicembre" ,"Gennaio" ,"Febbraio","Marzo","Aprile","Maggio","Giugno","Lug
> mesi=1:12
> names(mesi)=mesit
> m=names(mesi[mese])
> anno=year(as.POSIXlt(date(),format="%a %b %d %H:%M:%S %Y"))
> if(mese==1) anno=anno-1
> stringa=paste("Milano/", anno ,"/",m,"?format=csv",sep="")
```

Consideriamo la rilevazione della temperatura media giornaliera di Milano nel mese di Dicembre 2015. Scriviamo

```
>stringa="Milano/2015/Dicembre?format=csv"


> sito="http://www.ilmeteo.it/portale/archivio-meteo/"
> indirizzo=paste(sito,stringa,sep="")
> meteo=read.table(indirizzo,sep=";")


> dim(meteo)
[1] 32 15
> meteo[-1,3]
 [1] 6 4 4 3 6 6 7 6 5 3 2 3 6 6 6 8 6 5 5 6 7 8 8 7 6 3 2 1 1 6
[31] 5
Levels: 1 2 3 4 5 6 7 8 TMEDIA °C
```

A questo punto eliminiamo i livelli di `meteo` con il comando `as.vector` e consideriamo il risultato come numerico con

```
> as.numeric(as.vector(meteo[-1,3]))->Milano;
> Milano
 [1] 6 4 4 3 6 6 7 6 5 3 2 3 6 6 6 8 6 5 5 6 7 8 8 7 6 3 2 1 1 6
[31] 5
> quantile(Milano)
  0%  25%  50%  75% 100%
 1.0  3.5  6.0  6.0  8.0
```

L'ultimo comando in particolare ci fornisce minimo e massimo dei dati. Possiamo esaminare la serie temporale dei dati con i comandi

```
> plot(Milano,type="l",xlab="settembre 2010 a milano",ylab="temperatura media")
```

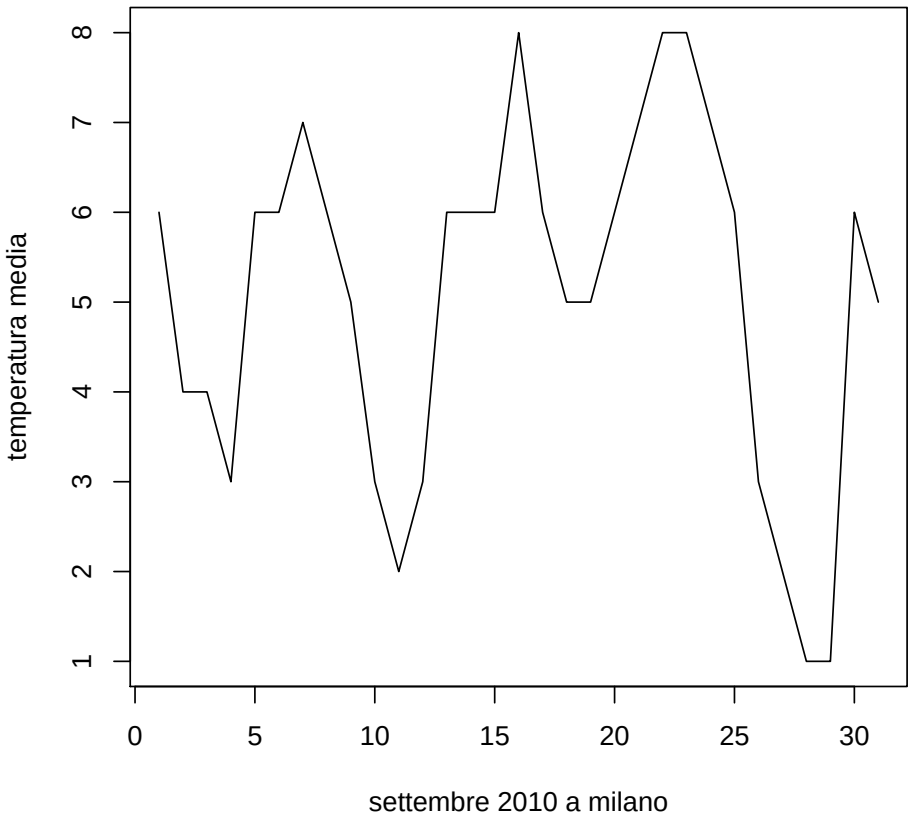

Figura 1.2: Andamento della temperatura a settembre 2010.

ottenendo la figura **??**

Raggruppiamo ora i dati in classi comprese tra due estremi che comprendano certamente tutti i dati, per esempio 13 e 23, decidendo di applicare un passo di 2 e vedere come si distribuiscono. Il comando `cut` associa a ciascun dato la classe di appartenenza selezionata in base ai punti di taglio.

```
> cut(Milano,breaks=seq(13.,23.,by=2))
 [1] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[13] <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA> <NA>
[25] <NA> <NA> <NA> <NA> <NA> <NA> <NA>
Levels: (13,15] (15,17] (17,19] (19,21] (21,23]
```

Il comando `table` conta i dati di ciascuna classe

```
> table(cut(Milano,breaks=seq(13.,23.,by=2)))
(13,15] (15,17] (17,19] (19,21] (21,23]
      0       0       0       0       0
```

Si noti che la suddivisione in classi prevede intervalli aperti a sinistra e chiusi a destra. Per suddividere in modo che gli intervalli siano chiusi a sinistra e aperti a destra si specifica il parametro `right=FALSE`. Possiamo anche usare il comando `seq` per specificare i tagli.

$$\texttt{table(cut(}\textit{variabile}\texttt{, breaks=seq(}\textit{estremo inf}\texttt{,}$$
$$\textit{estremo sup}\texttt{, by} = \textit{passo}\texttt{), right=FALSE))}$$

o in modo più generale

$$\texttt{table(cut(}\ \textit{variabile}\texttt{,}$$
$$\texttt{breaks=c(}\textit{estremo inferiore}, \ldots, \textit{estremo superiore}\texttt{))}$$

estremamente utile in quanto consente di raggruppare i dati in classi non necessariamente di ugual ampiezza.

```
> table(cut(Milano,breaks=c(12.5,14,15,16, 18,20,22.5),
+ right=F))
[12.5,14)   [14,15)   [15,16)   [16,18)   [18,20) [20,22.5)
        0         0         0         0         0         0
```

Volendo raggruppare in classi i dati delle precedenti uscite del dado possiamo scrivere

```
> table(cut(dadi100,breaks=0:6))
(0,1] (1,2] (2,3] (3,4] (4,5] (5,6]
   12    14    22    15    18    19
```

Se scegliamo di chiudere a sinistra gli intervalli

```
> table(cut(dadi100,breaks= 1:7,right=FALSE))
[1,2) [2,3) [3,4) [4,5) [5,6) [6,7)
   12    14    22    15    18    19
```

In questo caso la occorre prestare attenzione alla chiusura agli estremi degli intervalli

1.3.3 Areogrammi

Il comando generico per generare un istogramma è:

$$\texttt{hist}(\textit{variabile})$$

che segue però la struttura del comando `cut`. L'ampiezza di ciascuna classe salvo diversamente indicato è costante e decisa da R. È possibile variare tale condizione definendo una lista con i punti di taglio (*cutoff*) delle classi volute:

$$\texttt{hist}(\textit{variabile}, \texttt{c}(\textit{valore}_1, \textit{valore}_2, \ldots)) \tag{1.3}$$

Per esempio se `dadi100` rappresenta le solite 100 uscite del lancio del dado, il comando

```
> par(mfrow=c(1,2))
> hist(dadi100,breaks=seq(0.5,6.5,1),col="red")
> hist(dadi100,freq=FALSE,breaks=seq(0.5,6.5,1),col="blue")
```

genera l'istogramma (in rosso, a sinistra Figura **??**) con le frequenze assolute delle classi in ordinata. La sequenza dei punti di taglio è stata scelta in modo che i numeri interi da 1 a 6 siano al centro delle classi corrispondenti. Se invece volessimo creare un areogramma (ossia avere un tracciato per cui le aree siano pari alle frequenze relative) a partire dalle stesse uscite dovremo imporre il parametro `freq=FALSE` otterremo il pannello a destra (in blu) della figura (**??**). Avendo scelto classi di ampiezza costante i 2 grafici differiscono semplicemente per un cambio di scala sull'asse y.

In modo simile possiamo tracciare un areogramma dei dati nella variabile `milano`

```
> par(mfrow=c(1,2))
> hist(Milano, col="green",freq=FALSE,right=FALSE,
+ main="Cutoff automatici")
```

lasciando R libero di scegliere i punti di taglio (pannelli a sinistra della figura **??**) o scegliendoli a nostra volta (pannelli a destra della stessa figura **??**)

```
> hist(Milano,col="red",freq=FALSE,
+ breaks=unique(as.vector(quantile(Milano,seq(0,1,by=1/6)))),
+ main="Cutoff personalizzati")
```

Si noti la stabilità degli areogrammi rispetto ai cambi nella suddivisione.

1.3.4 Generazione di boxplot

Il `boxplot` è una rappresentazione grafica immediata della statistica dei 5 numeri e simultaneamente ci segnala eventuali punti discordanti o anomali, *outlier*. Il comando generico è:

$$\texttt{boxplot}(\textit{variabile}) \tag{1.4}$$

prendendo il vettore x contenente i risultati di 100 lanci otteniamo la figura **??** da cui si evince che il valore massimo dei dati è 6, il minimo è 1 e non ci sono punti anomali, per cui non vi sono dati anomali, altrimenti evidenziati da un pallino. Si legge inoltre il valore di mediana (4) primo quartile (2) e terzo quartile (5).

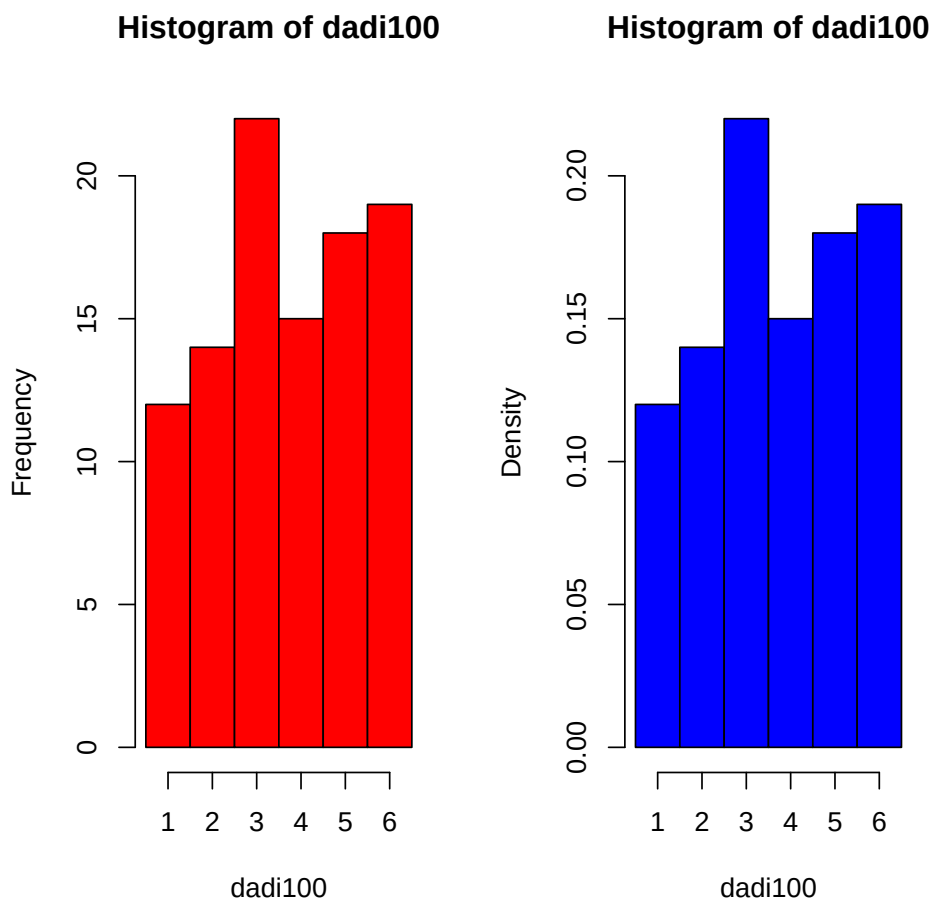

Figura 1.3: Diagramma a colonne e areogramma per il lancio di un dado.

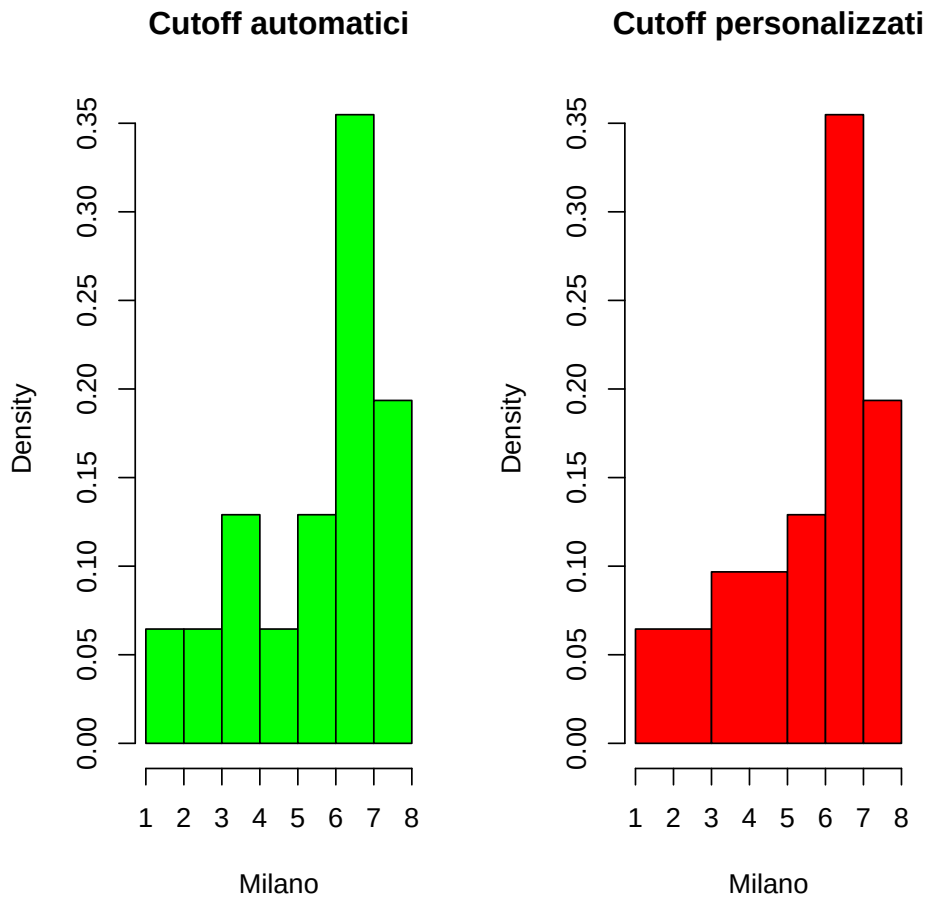

Figura 1.4: Areogramma dei dati della temperatura. Scelta automatica dei punti di taglio.

```
> boxplot(dadi100)
```

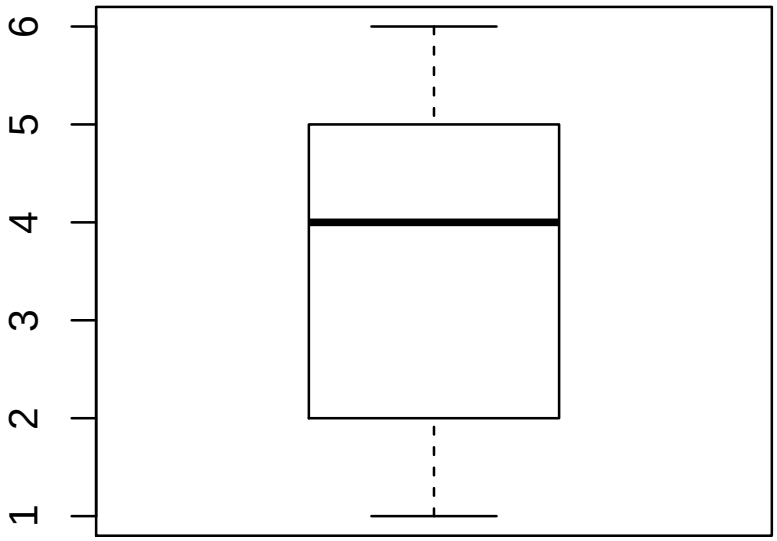

Figura 1.5: Boxplot dei risultati del lancio di un dado

1.3.5 Creazione di grafici a torta

Il comando `pie` consente, partendo da una tabella, di tracciare il diagramma a torta per una variabile nominale raggruppata in classi. Il comando è

`pie(table(`*variabile*`))`

ad esempio (facendo riferimento ai precedenti dati):

```
> pie(table(dadi100))
```

fornisce in uscita la Figura **??**

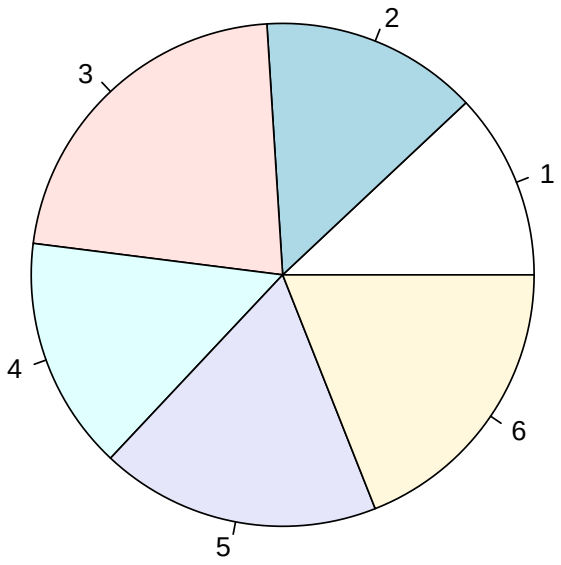

Figura 1.6: Diagramma a torta per il lancio di un dado equo.

Costruire una matrice contenente le coordinate di 50 punti nel rettangolo $[0,4] \times [0,2]$ in due dimensioni (generate utilizzando il generatore di numeri pseudocasuali). Produrre un grafico con due pannelli, dove il primo pannello è uno scatter-plot

1.4 Variabili doppie e rette di regressione

Supponiamo di misurare la concentrazione di acido lattico muscolare durante uno sforzo di 10 minuti,

```
> x<-tempo<-c(1,2,3,4,5,6,7,8,9,10)
> y<-concentrazione<-c(0.3,0.65,0.7,0.8,0.95,1.05,1.3,1.7,1.9,
+ 2.5)
```

Per analizzare questi dati conviene preliminarmente tracciarne un diagramma a dispersione. Possiamo inoltre determinare il coefficiente di correlazione lineare

```
> cor(x,y)
[1] 0.9620456
```

Per definire un modello di relazione lineare occorre usare il comando `lm` (*linear model*). Nella sua generica forma il comando è espresso come[1]

$$\texttt{lm}(y \sim x)$$

Otteniamo i valori di pendenza e intercetta.

Possiamo tracciare la retta di regressione con il comando `abline`.

```
> plot(x,y,pch=19,col="red")
> abline(lm(y~x),col="blue")
```

Per determinare la retta di regressione sulle y dobbiamo invertire x e y.

```
> lm(x~y)
Call:
lm(formula = x ~ y)

Coefficients:
(Intercept)            y
     0.3516       4.3446
```

[1] Per digitare la tilde $\sim$ su Mac premere ALT 5 su PC invece il tasto Alt Gr (attivazione del codice ASCII) e sul tastierino numerico digitare il numero 126. Lavorando su un portatile il tastierino numerico è spesso incorporato nella tastiera con colorazione blue dei tasti.

```
> plot(x,y)
```

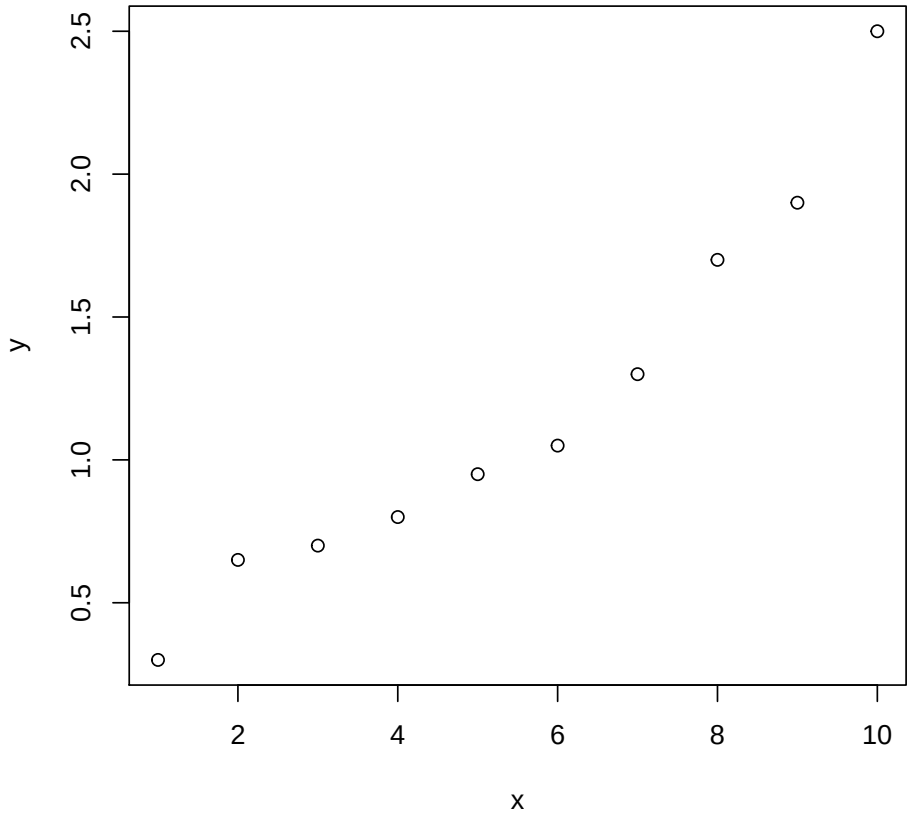

Figura 1.7: Diagramma a dispersione tempo/concentrazione.

```
> lm(x~y)$coefficients->coeff
> a=1/coeff[2];a;
        y
0.2301707
> b=-coeff[1]/coeff[2];
> abline(b,a,col="green")
```

In tal modo otteniamo il grafico **??**. Consideriamo ora il seguente *dataset* di mammiferi in

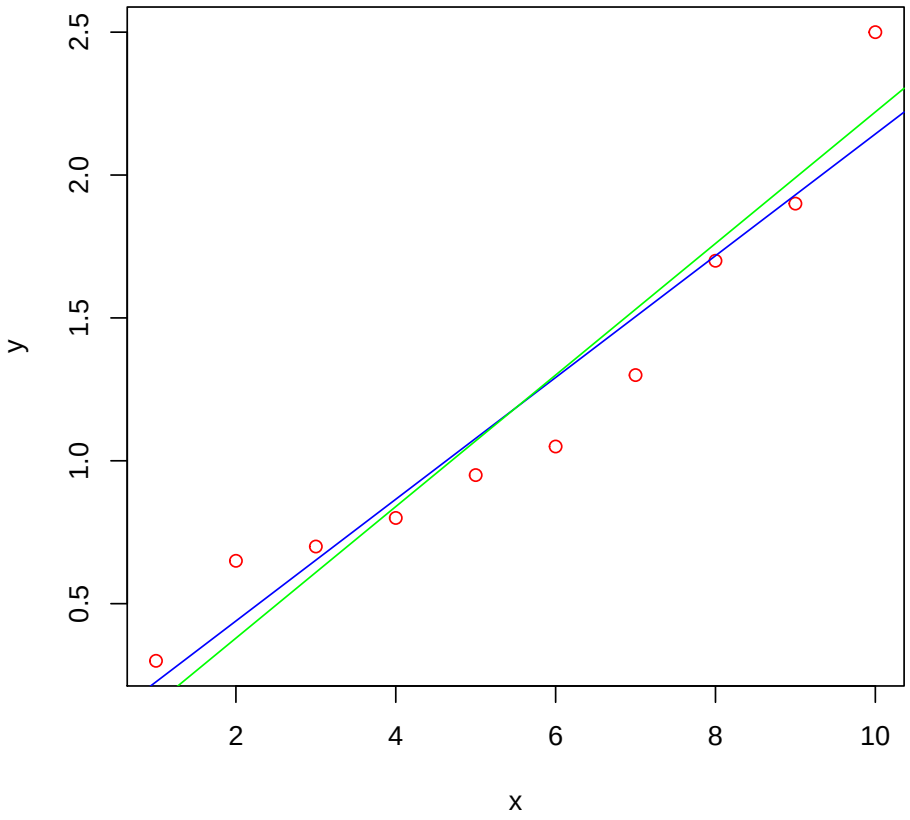

Figura 1.8: Rette di regressione. In blu R_x, in verde R_y.

cui le 2 variabili rappresentano le dimensioni del corpo e del cervello.

```
> library(MASS)
> mammals
```

	body	brain
Arctic fox	3.385	44.50
Owl monkey	0.480	15.50
Mountain beaver	1.350	8.10
Cow	465.000	423.00
Grey wolf	36.330	119.50
Goat	27.660	115.00
Roe deer	14.830	98.20
Guinea pig	1.040	5.50
Verbet	4.190	58.00
Chinchilla	0.425	6.40
Ground squirrel	0.101	4.00
Arctic ground squirrel	0.920	5.70
African giant pouched rat	1.000	6.60
Lesser short-tailed shrew	0.005	0.14
Star-nosed mole	0.060	1.00
Nine-banded armadillo	3.500	10.80
Tree hyrax	2.000	12.30
N.A. opossum	1.700	6.30
Asian elephant	2547.000	4603.00
Big brown bat	0.023	0.30
Donkey	187.100	419.00
Horse	521.000	655.00
European hedgehog	0.785	3.50
Patas monkey	10.000	115.00
Cat	3.300	25.60
Galago	0.200	5.00
Genet	1.410	17.50
Giraffe	529.000	680.00
Gorilla	207.000	406.00
Grey seal	85.000	325.00
Rock hyrax-a	0.750	12.30
Human	62.000	1320.00
African elephant	6654.000	5712.00
Water opossum	3.500	3.90
Rhesus monkey	6.800	179.00
Kangaroo	35.000	56.00
Yellow-bellied marmot	4.050	17.00
Golden hamster	0.120	1.00
Mouse	0.023	0.40
Little brown bat	0.010	0.25
Slow loris	1.400	12.50
Okapi	250.000	490.00

```
Rabbit                2.500   12.10
Sheep                55.500  175.00
Jaguar              100.000  157.00
Chimpanzee           52.160  440.00
Baboon               10.550  179.50
Desert hedgehog       0.550    2.40
Giant armadillo      60.000   81.00
Rock hyrax-b          3.600   21.00
Raccoon               4.288   39.20
Rat                   0.280    1.90
E. American mole      0.075    1.20
Mole rat              0.122    3.00
Musk shrew            0.048    0.33
Pig                 192.000  180.00
Echidna               3.000   25.00
Brazilian tapir     160.000  169.00
Tenrec                0.900    2.60
Phalanger             1.620   11.40
Tree shrew            0.104    2.50
Red fox               4.235   50.40
```

Per prima cosa tracciamo il grafico dei punti in scala non trasformata e, visto la compresenza di dati molto prossimi all'origine e di dati molto distanti in scala logaritmica (sia le x che le y vengono trasformate prendendone i logaritmi)

```
> par(mfrow=c(1,2))
> plot(mammals)
> plot(mammals,log="xy")
```

come in Figura **??**. Visti i risultati ottenuti usando la scala logaritmica tracciamo anche la corrispondente retta di regressione

```
> plot(log(mammals$brain)~log(mammals$body),col="BLUE",pch=19,type="p")
> abline(lm(log(mammals$brain)~ log(mammals$body)),col="red",lwd=3);
> uomo=which(rownames(mammals)=="Human")
> text(log(mammals[uomo ,1]),log(mammals[uomo ,2]),rownames(mammals)[uomo])
```

Si noti il comando `text(x,y, testo)` dove *x* e *y* e *testo* sono vettori di arbitraria lunghezza contenenti ascisse, ordinate e testo da inserire.

1.5 Distribuzioni in R

I nomi delle principali distribuzioni in R sono

```
> par(mfrow=c(1,2))
> plot(mammals)
> plot(mammals,log="xy")
```

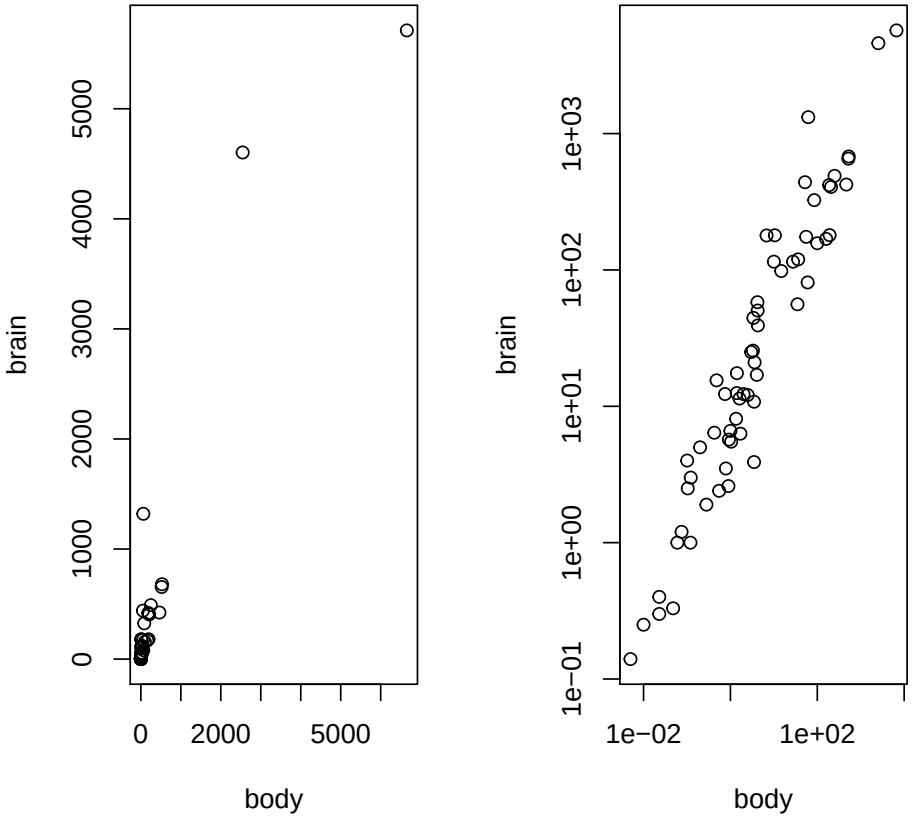

Figura 1.9: Diagramma a dispersione massa corporea/massa del cervello in scala normale ed in scala logaritmica.

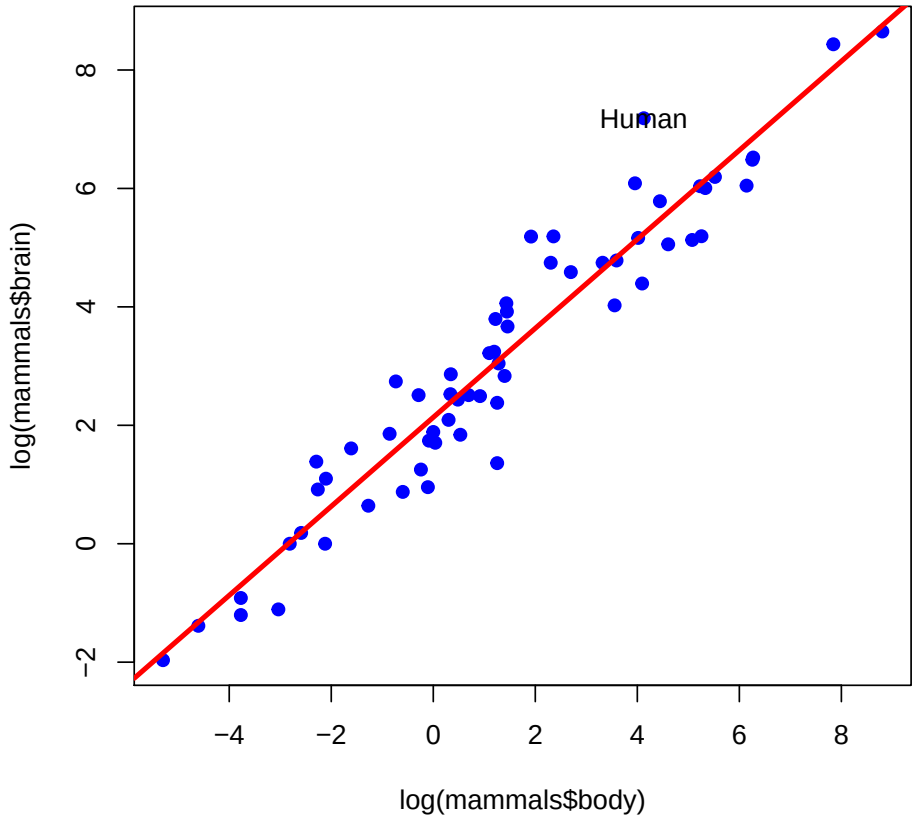

Figura 1.10: Retta di regressione. Dimensione del corpo e del cervello. Si noti la posizione dell'uomo.

`norm`	normale
`t`	Student
`chisq`	chi quadro
`f`	Fisher
`binom`	binomiale

A questi nomi possiamo aggiungere diversi prefissi

`d`	densità
`p`	primitiva
`q`	quantile
`r`	random

per caratterizzare diversi aspetti.

- **Esercizio**
  Figure **??** shows a scatterplot. Which of the following statements are correct?

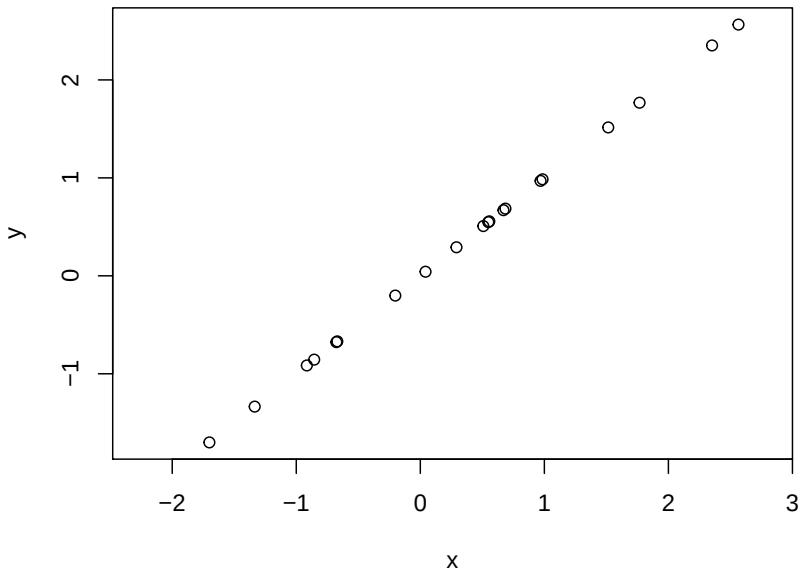

Figura 1.11: Scatterplot

(a) The standard deviation of Y is at least 6.

(b) The scatterplot is standardized.

(c) The mean of X is at most 5.

(d) For $X = 0.1$, Y can be expected to be about 0.2 .

(e) The absolute value of the correlation coefficient is at most 0.8.

Soluzione

(a) **False**: The standard deviation of Y is about equal to 1 and is therefore smaller than 6.

(b) **True**: X and Y have both mean 0 and variance 1.

(c) **True**: The mean of X is about equal to 0 and hence is smaller than 5.

(d) **False**: The regression line at $X = 0.1$ implies a value of about $Y = 0.9$.

(e) **False**: A strong association between the variables is given in the scatterplot. Hence the absolute value of the correlation coefficient is close to 1 and therefore larger than 0.8.

1.5.1 Distribuzione normale

La funzione `dnorm`

Come appena visto R indica con il nome `dnorm`, la densità normale o gaussiana. Essa accetta come parametri sia la media μ che la deviazione standard σ come è possibile verificare con il comando `formals` che ci fornisce gli argomenti di una funzione e gli eventuali valori preassegnati.

```
> formals(dnorm)
$x


$mean
[1] 0

$sd
[1] 1

$log
[1] FALSE
```

Se i parametri sono omessi `dnorm` rappresenta la densità normale standard con $\mu = 0$ e $\sigma = 1$. Il grafico (**??**) della gaussiana tra due estremi, ad esempio -2.5 e 2.5 si ottiene con il solito comando

```
> curve(dnorm,-2.5,2.5)
```

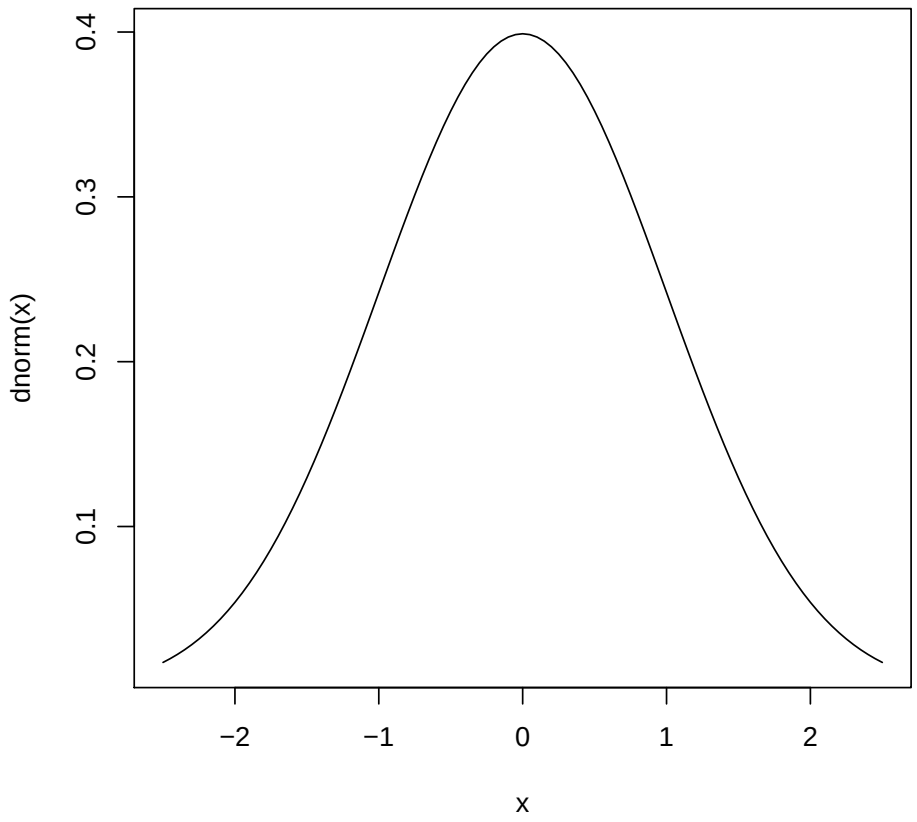

Figura 1.12: Grafico della normale standard nell'intervallo $[-2.5, 2.5]$.

Per visualizzare una gaussiana non standard, ad esempio con media $\mu = 1$, deviazione standard $\sigma = 1.5$, tra -3 e 3. scriveremo invece

```
> curve(dnorm(x,mean=1,sd=1.5),-3,3)
```

La funzione `pnorm`

La funzione `pnorm`(x) è definita come

$$\texttt{pnorm}(x) = \int_{-\infty}^{x} \texttt{dnorm}(s)ds$$

Ovviamente

$$\int_{a}^{b} \texttt{dnorm}(x)dx = \texttt{pnorm}(b) - \texttt{pnorm}(a)$$

e per avere l'area sottesa tra 3 e 5 basta scrivere:

```
> pnorm(5)-pnorm(3)
[1] 0.001349611
```

Per ottenere il valore dell'area tra 0 e x bisogna allora sottrarre `pnorm`(0)=0.5 all'area fornita dalla funzione. Per cui possiamo scrivere:

```
> pnorm(1)-0.5
[1] 0.3413447
```

La funzione `qnorm` e la tabella della densità di Gauss

Notiamo preliminarmente che se

$$\int_{-x}^{x} \texttt{dnorm}(s)ds = y$$

allora

$$\texttt{pnorm}(x) = \int_{-\infty}^{x} \texttt{dnorm}(s)ds = 1 - \frac{1-y}{2} = \frac{1+y}{2}$$

La funzione `qnorm` rappresenta la funzione inversa di `pnorm` quindi

$$x = \texttt{qnorm}\left(\frac{1+y}{2}\right)$$

In termini pratici possiamo introdurre la funzione

```
> u<-function(area) qnorm((1+area)/2)
```

fornisce fissato il livello di fiducia l'ascissa x tale che l'intervallo simmetrico $[-x, x]$ racchiuda un'area pari al lvello di fiducia. Per esempio

```
> u(0.95)
[1] 1.959964
```

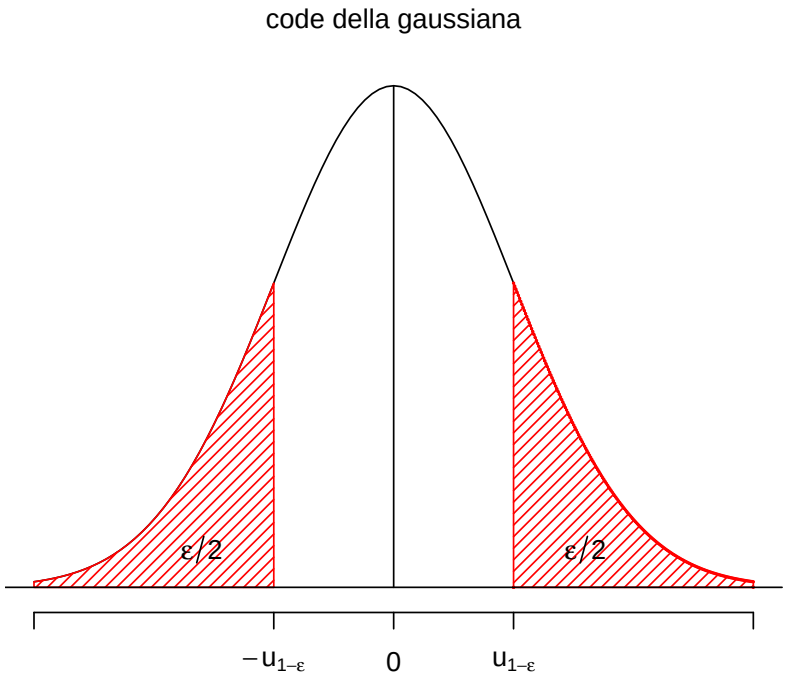

Figura 1.13: Code della distribuzione normale

La funzione `rnorm`

È possibile generare dei valori standardizzati casuali (media uguale a 0, deviazione standard pari a 1) che seguono la distribuzione normale standard. Basta semplicemente definire il numero di valori desiderati. Il comando nella sua espressione generale è:

$$\texttt{rnorm}(n, \texttt{mean} = \textit{valore}_1, \texttt{sd} = \textit{valore}_2) \tag{1.5}$$

Nel caso in cui volessimo una lista di 20 valori di una variabile normale con media assegnata 5 e deviazione standard 1 scriveremo :

```
> rnorm(20,mean=5,sd=1)
```

1.5.2 La distribuzione t di Student

In R la distribuzione di Student è indicata con la lettera `t`. Come per le altre densità si possono considerare le funzioni

dt	densità
pt	primitiva
qt	quantili
rt	generatore random

Il grafico della distribuzione di Student ad un certo numero `df` di gradi di libertà si ottiene con il comando

$$\texttt{curve(dt(x, df)}, a, b)$$

Tracciamo ad esempio un grafico tra -2 e 2 per una distribuzione a 10 gradi di libertà (vedi figura (**??**)):

Ricordiamo che la distribuzione di Student si usa in particolare nei casi in cui la deviazione standard della popolazione σ non è conosciuta e viene rimpiazzata dalla deviazione standard campionaria S, calcolata con un numero N di dati e quindi con $N-1$ gradi di libertà. Quando però il numero di dati si avvicina a 30 la curva di Student è praticamente sovrapposta a quella della distribuzione normale, come mostra il grafico (**??**):

1.5.3 Intervalli di confidenza e test di Student (dati non appaiati)

La funzione di R che esegue il test di Student nelle sue diverse forme è `t.test` Nella sua forma più semplice

```
> x=1:20;  t.test(x)
        One Sample t-test

data:  x
t = 7.9373, df = 19, p-value = 1.884e-07
```

```
> curve(dt(x,10),-2,2)
```

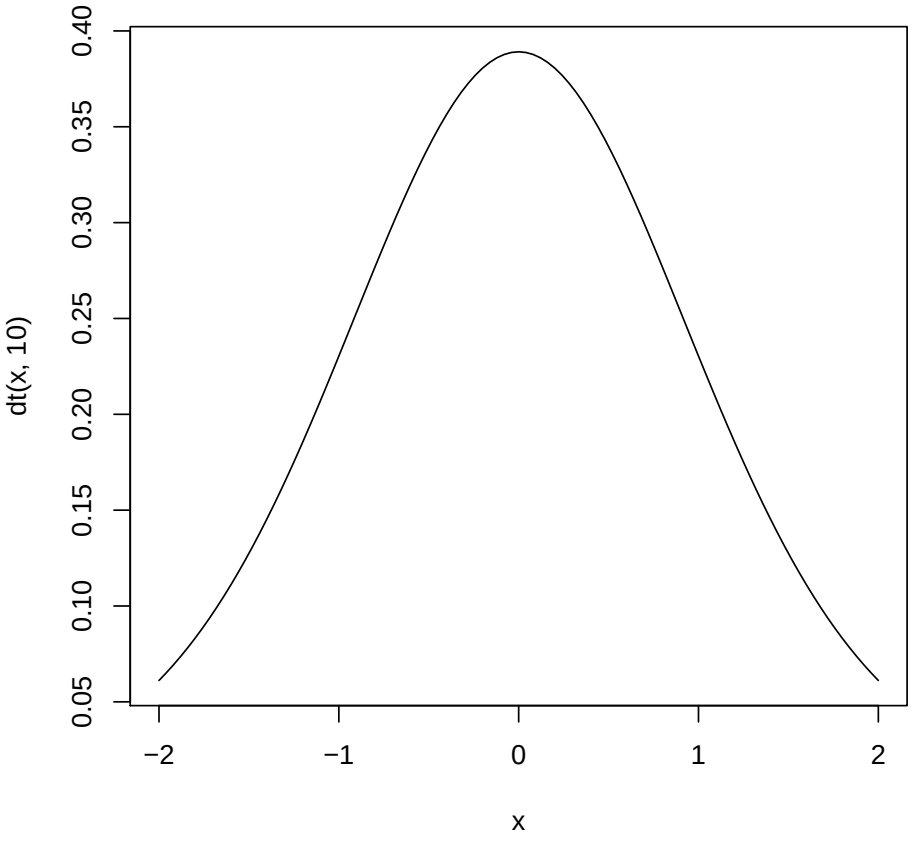

Figura 1.14: Grafico della distribuzione di Student a 10 gradi di libertà.

```
> curve(dnorm(x),-2,2,col=3)
> curve(dt(x,2),-2,2,col=1,add=T)
> curve(dt(x,25),-2,2,col=2,add=T)
> legend("topleft", c("df=2","df=25","normale"),pch=15,col=1:3);
```

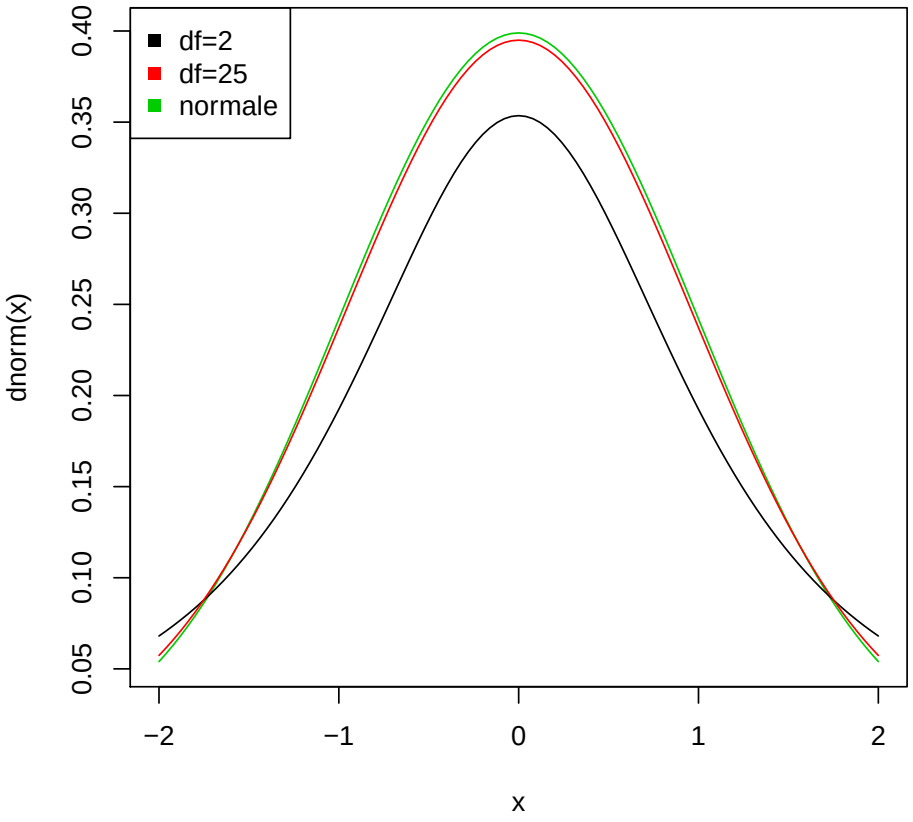

Figura 1.15: Grafico della distribuzione di Student a 10 gradi di libert.

```
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
  7.731189 13.268811
sample estimates:
mean of x
     10.5
```

In assenza di ipotesi R calcola il consuntivo

$$t = \frac{M_N(X) - \mu}{S_X}\sqrt{N}$$

assumendo che sia $\mu = 0$. Possiamo anche eseguire specificare l'ipotesi sul valore di μ:

```
> t.test(x,mu=7)

        One Sample t-test

data:  x
t = 2.6458, df = 19, p-value = 0.01595
alternative hypothesis: true mean is not equal to 7
95 percent confidence interval:
  7.731189 13.268811
sample estimates:
mean of x
     10.5
```

Possiamo infine specificare l'ipotesi alternativa. Per esempio se l'ipotesi alternativa è `"less"` il risultato del test cambia completamente.

```
> t.test(x,mu=7, alternative="less")

        One Sample t-test

data:  x
t = 2.6458, df = 19, p-value = 0.992
alternative hypothesis: true mean is less than 7
95 percent confidence interval:
     -Inf 12.78743
sample estimates:
mean of x
     10.5
```

In pratica ci viene fornito come p-value il valore dell'area sottesa dalla distribuzione di Student da $-\infty$ al valore di t se l'ipotesi alternativa è `"less"` e il valore dell'area sottesa dalla distribuzione di Student dal valore di t a $+\infty$ se l'ipotesi alternativa è `"greater"`

1.5.4 Test di Student per dati appaiati

Il test di Student per dati appaiati non è altro che un test di Student sulla differenza di 2 liste di dati di ugual lunghezza. Consideriamo ad esempio il confronto di 2 tecniche di misura applicate agli stessi campioni

```
> x<-c(1.46,2.22,2.84,1.97,1.13,2.35)
> y<-c(1.42,2.38,2.67,1.8,1.09,2.25)
```

Possiamo calcolare la differenza `x-y` ed applicare il test di Student oppure ottenere lo stesso risultato specificando l'opzione `paired=TRUE`

```
>  t.test(x,y,paired=TRUE)
        Paired t-test

data:  x and y
t = 1.2, df = 5, p-value = 0.2839
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.06852909  0.18852909
sample estimates:
mean of the differences
                   0.06
```

Il consuntivo `t` cade entro la regione di accettazione del test. È possibile specificare il livello di fiducia da utilizzare per il test di Student come:

$$\texttt{conf.level} = \textit{numero}$$

Il comando completo di tutti i parametri è quindi:

$$\texttt{t.test}(\textit{dati}_1, \textit{dati}_2, \texttt{paired=TRUE,conf.level} = \textit{valore})$$

Ad esempio eseguiamo un t-test per dati appaiati, tra $x = (1, 2, 3, 4)$ e $y = (3, 2, 4, 5)$ con *confidence level* di 0.85. Scriveremo

```
> t.test(1:4,5:2,paired=TRUE,conf.level=0.85)
```

Il consuntivo t cade fuori dalla regione di accettazione proposta.

1.6 Test χ^2 di indipendenza

Consideriamo il seguente *dataframe* che riporta le ambizioni di un gruppo di scolari americani

```
> read.table("http://lib.stat.cmu.edu/DASL/Datafiles/PopularKids.html",
+ skip=39,header=T,nrow=478,sep="\t")->kidinterest
```

```
  Gender Grade Age  Race Urban.Rural School   Goals
1    boy     5  11 White       Rural    Elm  Sports
2    boy     5  10 White       Rural    Elm Popular
3   girl     5  11 White       Rural    Elm Popular
4   girl     5  11 White       Rural    Elm Popular
5   girl     5  10 White       Rural    Elm Popular
6   girl     5  11 White       Rural    Elm Popular
  Grades Sports Looks Money
1      1      2     4     3
2      2      1     4     3
3      4      3     1     2
4      2      3     4     1
5      4      2     1     3
6      4      2     1     3
```

Nella tabella le colonne che ci interessano al momento sono quelle che riguardano il sesso, gli obiettivi (scelti tra successo scolastico, capacità sportiva e popolarità) e la provenienza (colonne 1, 5 e 7). Nelle colonne dalla 8 alla 11 sono messi in ordine di importanza per il conseguimento della popolarità voti, sport, aspetto esteriore e denaro.