Formulario di Statistica con

http://cran.r-project.org/other-docs.html

http://www.r-project.org/

Fabio Frascati[1]
Università degli Studi di Firenze
Firenze

Versione 2.3.1
Work in progress!

6 ottobre 2006

[1]Fabio Frascati, Laurea in Statistica e Scienze Economiche conseguita presso l'Università degli Studi di Firenze, fabiofrascati@yahoo.it

Indice

Parte I

Background

Capitolo 1

Funzioni matematiche

1.1 Operatori matematici

+

- **Package:** `base`
- **Significato:** addizione
- **Esempio:**

```
> 1+2
[1] 3

> x
[1] 1 2 3 4 5
> y
[1] 1.2 3.4 5.2 3.5 7.8
> x+y
[1]  2.2  5.4  8.2  7.5 12.8

> x
[1] 1 2 3 4 5
> x+10
[1] 11 12 13 14 15
```

−

- **Package:** `base`
- **Significato:** sottrazione
- **Esempio:**

```
> 1.2-6.7
[1] -5.5

> --3
[1] 3

> Inf-Inf
[1] NaN
> # NaN = Not a Number

> x
[1] 1 2 3 4 5
> y
[1] 1.2 3.4 5.2 3.5 7.8
> x-y
```

```
[1] -0.2 -1.4 -2.2  0.5 -2.8

> x
[1] 1 2 3 4 5
> x-10
[1] -9 -8 -7 -6 -5
```

*

- **Package:** base
- **Significato:** moltiplicazione
- **Esempio:**

```
> 2.3*4
[1] 9.2

> x
[1] 1 2 3 4 5 6 7
> y
[1] -3.2 -2.2 -1.2 -0.2  0.8  1.8  2.8
> x*y
[1] -3.2 -4.4 -3.6 -0.8  4.0 10.8 19.6
```

/

- **Package:** base
- **Significato:** divisione
- **Esempio:**

```
> 21/7
[1] 3

> 2/0
[1] Inf

> -1/0
[1] -Inf

> 0/0
[1] NaN
> # NaN = Not a Number

> Inf/Inf
[1] NaN

> Inf/0
[1] Inf

> -Inf/0
[1] -Inf

> x
[1] 1 2 3 4 5 6 7
> y
[1] -3.2 -2.2 -1.2 -0.2  0.8  1.8  2.8
> y/x
[1] -3.20 -1.10 -0.40 -0.05  0.16  0.30  0.40
```

**

- **Package:** base
- **Significato:** elevamento a potenza
- **Esempio:**

```
> 2**4
[1] 16

> x
[1] 1 2 3 4
> y
[1] -3.2 -2.2 -1.2 -0.2
> y**x
[1] -3.2000  4.8400 -1.7280  0.0016
```

^

- **Package:** base
- **Significato:** elevamento a potenza
- **Esempio:**

```
> 2^4
[1] 16

> x
[1] 1 2 3 4
> y
[1] -3.2 -2.2 -1.2 -0.2
> y^x
[1] -3.2000  4.8400 -1.7280  0.0016
```

%/%

- **Package:** base
- **Significato:** quoziente intero della divisione
- **Esempio:**

```
> 22.6%/%3.4
[1] 6
> # 22.6 = 3.4 * 6 + 2.2

> 23%/%3
[1] 7
> # 23 = 3 * 7 + 2
```

%%

- **Package:** base
- **Significato:** resto della divisione intera (modulo)
- **Esempio:**

```
> 22.6%%3.4
[1] 2.2
> # 22.6 = 3.4 * 6 + 2.2

> 23%%3
[1] 2
> # 23 = 3 * 7 + 2
```

1.2 Operatori relazionali

$\boxed{<}$

- **Package:** `base`
- **Significato:** minore
- **Esempio:**

```
> 1<2
[1] TRUE

> x
[1] 0.11 1.20 2.30 4.50
> x<2.4
[1]  TRUE  TRUE  TRUE FALSE
```

$\boxed{>}$

- **Package:** `base`
- **Significato:** maggiore
- **Esempio:**

```
> 3>1.2
[1] TRUE

> x
[1] 0.11 1.20 2.30 4.50
> x>2.4
[1] FALSE FALSE FALSE  TRUE
```

$\boxed{<=}$

- **Package:** `base`
- **Significato:** minore od uguale
- **Esempio:**

```
> 3.4<=8.5
[1] TRUE

> x
[1] 0.11 1.20 2.30 4.50
> x<=2.4
[1]  TRUE  TRUE  TRUE FALSE
```

`>=`

- **Package:** `base`
- **Significato:** maggiore od uguale
- **Esempio:**

```
> 3.4>=5.4
[1] FALSE

> x
[1] 0.11 1.20 2.30 5.40
> x>=5.4
[1] FALSE FALSE FALSE  TRUE
```

`!=`

- **Package:** `base`
- **Significato:** diverso
- **Esempio:**

```
> 2!=3
[1] TRUE

> x
[1] 0.11 1.20 2.30 5.40
> x!=5.4
[1]  TRUE  TRUE  TRUE FALSE
```

`==`

- **Package:** `base`
- **Significato:** uguale
- **Esempio:**

```
> 4==4
[1] TRUE

> x
[1] 0.11 1.20 2.30 5.40
> x==5.4
[1] FALSE FALSE FALSE  TRUE

> T==1
[1] TRUE

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

1.3 Operatori logici

`&`

- **Package:** `base`

- **Significato:** AND termine a termine
- **Esempio:**

```
> 1&5
[1] TRUE

> x
[1] 0.11 1.20 2.30 4.50 0.00
> x&3
[1]  TRUE  TRUE  TRUE  TRUE FALSE
```

&&

- **Package:** `base`
- **Significato:** AND si arresta al primo elemento che soddisfa la condizione
- **Esempio:**

```
> 1&&5
[1] TRUE

> x
[1] 0.11 1.20 2.30 4.50 0.00
> x&&3
[1] TRUE

> x
[1] 0.0 1.2 2.3 4.5 0.0
> x&&3
[1] FALSE
```

|

- **Package:** `base`
- **Significato:** OR termine a termine
- **Esempio:**

```
> 5|0
[1] TRUE

> x
[1] 0.11 1.20 2.30 4.50 0.00
> x|0
[1]  TRUE  TRUE  TRUE  TRUE FALSE
```

||

- **Package:** `base`
- **Significato:** OR si arresta al primo elemento che soddisfa la condizione
- **Esempio:**

```
> 5||0
[1] TRUE

> x
[1] 0.11 1.20 2.30 4.50 0.00
> x||3
[1] TRUE

> x
[1] 0.0 1.2 2.3 4.5 0.0
> x||0
[1] FALSE
```

xor()

- **Package:** `base`
- **Significato:** EXCLUSIVE OR termine a termine
- **Esempio:**

```
> xor(4,5)
[1] FALSE

> x
[1] 0.11 1.20 2.30 4.50 0.00
> xor(x,3)
[1] FALSE FALSE FALSE FALSE  TRUE
```

!

- **Package:** `base`
- **Significato:** NOT
- **Esempio:**

```
> !8
[1] FALSE

> x
[1] 0.11 1.20 2.30 4.50 0.00
> !x
[1] FALSE FALSE FALSE FALSE  TRUE
```

1.4 Funzioni di base

sum()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** somma
- **Formula:**

$$\sum_{i=1}^{n} x_i$$

- **Esempio:**

```
> x
[1] 1.2 2.0 3.0
> 1.2+2+3
[1] 6.2
> sum(x)
[1] 6.2

> x
[1] 1.2 3.4 5.1 5.6 7.8
> 1.2+3.4+5.1+5.6+7.8
[1] 23.1
> sum(x)
[1] 23.1
```

prod()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** prodotto
- **Formula:**

$$\prod_{i=1}^{n} x_i$$

- **Esempio:**

```
> x
[1] 1 2 3.2
> 1*2*3.2
[1] 6.4
> prod(x)
[1] 6.4

> x
[1] 1.2 3.4 5.1 5.6 7.8
> 1.2*3.4*5.1*5.6*7.8
[1] 908.8934
> prod(x)
[1] 908.8934
```

abs()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** valore assoluto
- **Formula:**

$$|\,x\,| = \begin{cases} x & \text{se } x > 0 \\ 0 & \text{se } x = 0 \\ -x & \text{se } x < 0 \end{cases}$$

- **Esempio:**

```
> abs(x=1.3)
[1] 1.3

> abs(x=0)
[1] 0

> abs(x=-2.3)
[1] 2.3
```

- **Osservazioni:** Equivale alla funzione `Mod()`.

sign()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** segno
- **Formula:**

$$\text{sign}(x) = \begin{cases} 1 & \text{se } x > 0 \\ 0 & \text{se } x = 0 \\ -1 & \text{se } x < 0 \end{cases}$$

- **Esempio:**

```
> sign(x=1.2)
[1] 1

> sign(x=0)
[1] 0

> sign(x=-1.2)
[1] -1
```

sqrt()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $x > 0$

- **Significato:** radice quadrata
- **Formula:**

$$\sqrt{x}$$

- **Esempio:**

```
> sqrt(x=2)
[1] 1.414214

> sqrt(x=3.5)
[1] 1.870829

> sqrt(x=-9)
[1] NaN
Warning message:
Si è prodotto un NaN in: sqrt(-9)
> sqrt(x=-9+0i)
[1] 0+3i
```

1.5 Funzioni insiemistiche

union()

- **Package:** `base`
- **Parametri:**
  - `x` vettore alfanumerico di dimensione n
  - `y` vettore alfanumerico di dimensione m
- **Significato:** unione
- **Formula:**

$$x \cup y$$

- **Esempio:**

```
> x
[1]  1  2  3  4  5  6  7  8  9 10
> y
[1]  1  2  6 11
> union(x,y)
[1]  1  2  3  4  5  6  7  8  9 10 11

> x
[1] "a" "b" "c" "d" "e" "f" "g"
> y
[1] "a" "e" "f" "h"
> union(x,y)
[1] "a" "b" "c" "d" "e" "f" "g" "h"
```

intersect()

- **Package:** `base`
- **Parametri:**
  - `x` vettore alfanumerico di dimensione n
  - `y` vettore alfanumerico di dimensione m
- **Significato:** intersezione
- **Formula:**

$$x \cap y$$

- **Esempio:**

```
> x
[1]  1  2  3  4  5  6  7  8  9 10
> y
[1]  1  2  6 11
> intersect(x,y)
[1] 1 2 6

> x
[1] "a" "b" "c" "d" "e" "f" "g"
> y
[1] "a" "e" "f" "h"
> intersect(x,y)
[1] "a" "e" "f"
```

setdiff()

- **Package:** `base`
- **Parametri:**

  `x` vettore alfanumerico di dimensione n

  `y` vettore alfanumerico di dimensione m

- **Significato:** differenza
- **Formula:**

$$x \setminus y$$

- **Esempio:**

```
> x
[1]  1  2  3  4  5  6  7  8  9 10
> y
[1]  1  2  6 11
> setdiff(x,y)
[1]  3  4  5  7  8  9 10

> x
[1] "a" "b" "c" "d" "e" "f" "g"
> y
[1] "a" "e" "f" "h"
> setdiff(x,y)
[1] "b" "c" "d" "g"
```

is.element()

- **Package:** `base`
- **Parametri:**

  `el` valore x alfanumerico

  `set` vettore y alfanumerico di dimensione n

- **Significato:** appartenenza di x all'insieme y
- **Formula:**

$$x \in y$$

- **Esempio:**

```
> x
[1] 2
> y
[1]  1  2  6 11
> is.element(el=x,set=y)
[1] TRUE

> x
[1] 3
> y
[1]  1  2  6 11
> is.element(el=x,set=y)
[1] FALSE

> x
[1] "d"
> y
[1] "a" "b" "c" "d" "e" "f" "g"
```

```
> is.element(el=x,set=y)
[1] TRUE

> x
[1] "h"
> y
[1] "a" "b" "c" "d" "e" "f" "g"
> is.element(el=x,set=y)
[1] FALSE
```

%in%

- **Package:** `base`
- **Parametri:**

  `x` valore alfanumerico

  `y` vettore alfanumerico di dimensione n

- **Significato:** appartenenza di x all'insieme y
- **Formula:**

$$x \in y$$

- **Esempio:**

```
> x
[1] 2
> y
[1]  1  2  6 11
> x%in%y
[1] TRUE

> x
[1] 3
> y
[1]  1  2  6 11
> x%in%y
[1] FALSE

> x
[1] "d"
> y
[1] "a" "b" "c" "d" "e" "f" "g"
> x%in%y
[1] TRUE

> x
[1] "h"
> y
[1] "a" "b" "c" "d" "e" "f" "g"
> x%in%y
[1] FALSE
```

setequal()

- **Package:** `base`
- **Parametri:**

  `x` vettore alfanumerico di dimensione n

`y` vettore alfanumerico di dimensione m

- **Significato:** uguaglianza
- **Formula:**

$$x = y \Leftrightarrow \begin{cases} x \subseteq y \\ y \subseteq x \end{cases}$$

- **Esempio:**

```
> x
[1]  1  4  5  6  8 77
> y
[1]  1  1  1  4  5  6  8 77
> setequal(x,y)
[1] TRUE

> x
[1] "a" "b"
> y
[1] "a" "b" "a" "b" "a" "b" "a"
> setequal(x,y)
[1] TRUE
```

1.6 Funzioni indice

which()

- **Package:** `base`
- **Parametri:**

`x` vettore numerico di dimensione n

- **Significato:** indici degli elementi di x che soddisfano ad una condizione fissata
- **Esempio:**

```
> x
[1]  1.2  4.5 -1.3  4.5
> which(x>2)
[1] 2 4

> x
[1]  1.2  4.5 -1.3  4.5
> which((x>=-1)&(x<5))
[1] 1 2 4

> x
[1]  1.2  4.5 -1.3  4.5
> which((x>=3.6)|(x< -1.6))
[1] 2 4
```

which.min()

- **Package:** `base`
- **Parametri:**

`x` vettore numerico di dimensione n

- **Significato:** indice del primo elemento minimo di x

- **Esempio:**

```
> x
[1] 1.2 1.0 2.3 4.0 1.0 4.0
> which.min(x)
[1] 2

> x
[1]  1.2  4.5 -1.3  4.5
> which.min(x)
[1] 3
```

which.max()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** indice del primo elemento massimo di x
- **Esempio:**

```
> x
[1] 1.2 1.0 2.3 4.0 1.0 4.0
> which.max(x)
[1] 4

> x
[1]  1.2  4.5 -1.3  4.5
> which.max(x)
[1] 2
```

1.7 Funzioni combinatorie

choose()

- **Package:** `base`
- **Parametri:**

  `n` valore naturale

  `k` valore naturale tale che $0 \leq k \leq n$

- **Significato:** coefficiente binomiale
- **Formula:**

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

- **Esempio:**

```
> n<-10
> k<-3
> prod(1:n)/(prod(1:k)*prod(1:(n-k)))
[1] 120
> choose(n=10,k=3)
[1] 120

> n<-8
> k<-5
```

```
> prod(1:n)/(prod(1:k)*prod(1:(n-k)))
[1] 56
> choose(n=8,k=5)
[1] 56
```

lchoose()

- **Package:** `base`
- **Parametri:**

  `n` valore naturale

  `k` valore naturale tale che $0 \leq k \leq n$

- **Significato:** logaritmo naturale del coefficiente binomiale
- **Formula:**

$$\log \binom{n}{k}$$

- **Esempio:**

```
> n<-10
> k<-3
> log(prod(1:n)/(prod(1:k)*prod(1:(n-k))))
[1] 4.787492
> lchoose(n=10,k=3)
[1] 4.787492

> n<-8
> k<-5
> log(prod(1:n)/(prod(1:k)*prod(1:(n-k))))
[1] 4.025352
> lchoose(n=8,k=5)
[1] 4.025352
```

factorial()

- **Package:** `base`
- **Parametri:**

  `x` valore naturale

- **Significato:** fattoriale
- **Formula:**

$$x\,!$$

- **Esempio:**

```
> x<-4
> prod(1:x)
[1] 24
> factorial(x=4)
[1] 24

> x<-6
> prod(1:x)
[1] 720
> factorial(x=6)
[1] 720
```

lfactorial()

- **Package:** `base`
- **Parametri:**

  `x` valore naturale

- **Significato:** logaritmo del fattoriale in base e
- **Formula:**

$$\log(x\,!)$$

- **Esempio:**

```
> x<-4
> log(prod(1:x))
[1] 3.178054
> lfactorial(x=4)
[1] 3.178054

> x<-6
> log(prod(1:x))
[1] 6.579251
> lfactorial(x=6)
[1] 6.579251
```

1.8 Funzioni trigonometriche

sin()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** seno
- **Formula:**

$$\sin(x)$$

- **Esempio:**

```
> sin(x=1.2)
[1] 0.932039

> sin(x=pi)
[1] 1.224606e-16
```

cos()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** coseno
- **Formula:**

$$\cos(x)$$

- **Esempio:**

```
> cos(x=1.2)
[1] 0.3623578

> cos(x=pi/2)
[1] 6.123032e-17
```

tan()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** tangente
- **Formula:**

$$\tan(x)$$

- **Esempio:**

```
> tan(x=1.2)
[1] 2.572152

> tan(x=pi)
[1] -1.224606e-16
```

asin()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $|x| \leq 1$

- **Significato:** arcoseno di x, espresso in radianti nell'intervallo tra $-\pi / 2$ e $\pi / 2$
- **Formula:**

$$\arcsin(x)$$

- **Esempio:**

```
> asin(x=0.9)
[1] 1.119770

> asin(x=-1)
[1] -1.570796
```

acos()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $|x| \leq 1$

- **Significato:** arcocoseno di x, espresso in radianti nell'intervallo tra 0 e π
- **Formula:**

$$\arccos(x)$$

- **Esempio:**

```
> acos(x=0.9)
[1] 0.4510268

> acos(x=-1)
[1] 3.141593
```

atan()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** arcotangente di x, espressa in radianti nell'intervallo tra $-\pi\,/\,2$ e $\pi\,/\,2$
- **Formula:**

$$\arctan(x)$$

- **Esempio:**

```
> atan(x=0.9)
[1] 0.7328151

> atan(x=-34)
[1] -1.541393
```

atan2()

- **Package:** `base`
- **Parametri:**

  `y` valore numerico di ordinata

  `x` valore numerico di ascissa

- **Significato:** arcotangente in radianti dalle coordinate x e y specificate, nell'intervallo tra $-\pi$ e π
- **Formula:**

$$\arctan(x)$$

- **Esempio:**

```
> atan2(y=-2,x=0.9)
[1] -1.147942

> atan2(y=-1,x=-1)
[1] -2.356194
```

sinh()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** seno iperbolico
- **Formula:**

$$\sinh(x) = \frac{e^x - e^{-x}}{2}$$

- **Esempio:**

```
> x<-2.45
> (exp(x)-exp(-x))/2
[1] 5.751027
> sinh(x=2.45)
[1] 5.751027

> x<-3.7
> (exp(x)-exp(-x))/2
[1] 20.21129
> sinh(x=3.7)
[1] 20.21129
```

cosh()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** coseno iperbolico
- **Formula:**

$$\cosh(x) = \frac{e^x + e^{-x}}{2}$$

- **Esempio:**

```
> x<-2.45
> (exp(x)+exp(-x))/2
[1] 5.83732
> cosh(x=2.45)
[1] 5.83732

> x<-3.7
> (exp(x)+exp(-x))/2
[1] 20.23601
> cosh(x=3.7)
[1] 20.23601
```

tanh()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** tangente iperbolica
- **Formula:**

$$\tanh(x) = \frac{e^{2\,x} - 1}{e^{2\,x} + 1}$$

- **Esempio:**

```
> x<-2.45
> (exp(2*x)-1)/(exp(2*x)+1)
[1] 0.985217
> tanh(x=2.45)
[1] 0.985217
```

```
> x<-3.7
> (exp(2*x)-1)/(exp(2*x)+1)
[1] 0.9987782
> tanh(x=3.7)
[1] 0.9987782
```

asinh()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** inversa seno iperbolico
- **Formula:**

$$\operatorname{arcsinh}(x)$$

- **Esempio:**

```
> asinh(x=2.45)
[1] 1.628500

> asinh(x=3.7)
[1] 2.019261
```

acosh()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $x \geq 1$

- **Significato:** inversa coseno iperbolico
- **Formula:**

$$\operatorname{arccosh}(x)$$

- **Esempio:**

```
> acosh(x=2.45)
[1] 1.544713

> acosh(x=3.7)
[1] 1.982697
```

atanh()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $|x| < 1$

- **Significato:** inversa tangente iperbolica
- **Formula:**

$$\operatorname{arctanh}(x)$$

- **Esempio:**

```
> atanh(x=0.45)
[1] 0.4847003

> atanh(x=0.7)
[1] 0.8673005
```

1.9 Funzioni esponenziali e logaritmiche

exp()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** esponenziale
- **Formula:**

$$e^x$$

- **Esempio:**

```
> exp(x=1.2)
[1] 3.320117

> exp(x=0)
[1] 1
```

expm1()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** esponenziale
- **Formula:**

$$e^x - 1$$

- **Esempio:**

```
> x<-1.2
> exp(x)-1
[1] 2.320117
> expm1(x=1.2)
[1] 2.320117

> x<-0
> exp(x)-1
[1] 0
> expm1(x=0)
[1] 0
```

log2()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $x > 0$

- **Significato:** logaritmo di x in base 2
- **Formula:**

$$\log_2(x)$$

- **Esempio:**

```
> log2(x=1.2)
[1] 0.2630344

> log2(x=8)
[1] 3
```

log10()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $x > 0$

- **Significato:** logaritmo di x in base 10
- **Formula:**

$$\log_{10}(x)$$

- **Esempio:**

```
> log10(x=1.2)
[1] 0.07918125

> log10(x=1000)
[1] 3
```

log()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $x > 0$

  `base` il valore b tale che $b > 0$

- **Significato:** logaritmo di x in base b
- **Formula:**

$$\log_b(x)$$

- **Esempio:**

```
> log(x=2,base=4)
[1] 0.5

> log(x=8,base=2)
[1] 3

> log(x=0,base=10)
[1] -Inf

> log(x=100,base=-10)
[1] NaN
Warning message:
Si è prodotto un NaN in: log(x, base)
```

logb()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $x > 0$

  `base` il valore b tale che $b > 0$

- **Significato:** logaritmo di x in base b
- **Formula:**

$$\log_b(x)$$

- **Esempio:**

```
> logb(x=2,base=4)
[1] 0.5

> logb(x=8,base=2)
[1] 3
```

log1p()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $x > -1$

- **Significato:** logaritmo di x in base e
- **Formula:**

$$\log(x+1)$$

- **Esempio:**

```
> x<-2.3
> log(x+1)
[1] 1.193922
> log1p(x=2.3)
[1] 1.193922

> x<-8
> log(x+1)
[1] 2.197225
> log1p(x=8)
[1] 2.197225
```

1.10 Funzioni di successione

:

- **Package:** `base`
- **Significato:** successione con intervallo unitario
- **Esempio:**

```
> 1:10
[1]  1  2  3  4  5  6  7  8  9 10
> 1:10.2
[1]  1  2  3  4  5  6  7  8  9 10
> 1.1:10.2
[1]  1.1  2.1  3.1  4.1  5.1  6.1  7.1  8.1  9.1 10.1

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

rep()

- **Package:** `base`
- **Parametri:**

  `x` vettore alfanumerico di dimensione n

  `times` ogni elemento del vettore viene ripetuto lo stesso numero *times* di volte

  `length.out` dimensione del vettore risultato

  `each` ogni elemento del vettore viene ripetuto *each* volte

- **Significato:** replicazioni
- **Esempio:**

```
> rep(x=2,times=5)
[1] 2 2 2 2 2

> rep(x=c(1,2,3),times=5)
[1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3

> rep(x=c(1,2,3),times=c(1,2,3))
[1] 1 2 2 3 3 3

> rep(x=c(1,2,3),each=2)
[1] 1 1 2 2 3 3

> rep(x=c(1,2,3),length.out=7)
[1] 1 2 3 1 2 3 1

> rep(x=T,times=5)
[1] TRUE TRUE TRUE TRUE TRUE

> rep(x=c(1,2,3,4),times=2,each=3)
[1] 1 1 1 2 2 2 3 3 3 4 4 4 1 1 1 2 2 2 3 3 3 4 4 4
```

rep.int()

- **Package:** `base`
- **Parametri:**

  `x` vettore alfanumerico di dimensione n

  `times` ogni elemento del vettore viene ripetuto lo stesso numero *times* di volte

- **Significato:** replicazioni
- **Esempio:**

```
> rep.int(x=2,times=5)
[1] 2 2 2 2 2

> rep.int(x=c(1,2,3),times=5)
[1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3

> rep.int(x=c(1,2,3),times=c(1,2,3))
[1] 1 2 2 3 3 3

> rep(x=T,times=5)
[1] TRUE TRUE TRUE TRUE TRUE
```

sequence()

- **Package:** `base`
- **Parametri:**

  `nvec` vettore numerico x di valori naturali di dimensione n

- **Significato:** serie di sequenze di interi dove ciascuna sequenza termina con i numeri passati come argomento
- **Esempio:**

```
> n1<-2
> n2<-5
> c(1:n1,1:n2)
[1] 1 2 1 2 3 4 5
> sequence(nvec=c(2,5))
[1] 1 2 1 2 3 4 5

> n1<-6
> n2<-3
> c(1:n1,1:n2)
[1] 1 2 3 4 5 6 1 2 3
> sequence(nvec=c(6,3))
[1] 1 2 3 4 5 6 1 2 3
```

seq()

- **Package:** `base`
- **Parametri:**

  `from` punto di partenza

  `to` punto di arrivo

  `by` passo

  `length.out` dimensione

`along.with` vettore di dimensione n per creare la sequenza di valori naturali 1, 2, ..., n

- **Significato:** successione
- **Esempio:**

```
> seq(from=1,to=3.4,by=0.4)
[1] 1.0 1.4 1.8 2.2 2.6 3.0 3.4

> seq(from=1,to=3.4,length.out=5)
[1] 1.0 1.6 2.2 2.8 3.4

> seq(from=3.4,to=1,length.out=5)
[1] 3.4 2.8 2.2 1.6 1.0

> x
[1]  1  2  6 11
> n<-length(x)
> n
[1] 4
> 1:n
[1] 1 2 3 4
> seq(along.with=x)
[1] 1 2 3 4

> seq(from=5,by=-1,along.with=1:6)
[1] 5 4 3 2 1 0

> seq(from=8)
[1] 1 2 3 4 5 6 7 8

> seq(from=-8)
[1]  1  0 -1 -2 -3 -4 -5 -6 -7 -8
```

1.11 Funzioni di ordinamento

sort()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `decreasing = T / F` decremento oppure incremento

  `index.return = T / F` vettore indici ordinati

- **Significato:** ordinamento crescente oppure decrescente
- **Output:**

  `x` vettore ordinato

  `ix` vettore indici ordinati

- **Formula:**

  `x`

  `decreasing = T`

  $$x_{(n)}, x_{(n-1)}, \ldots, x_{(1)}$$

  `decreasing = F`

  $$x_{(1)}, x_{(2)}, \ldots, x_{(n)}$$

- **Esempio:**

```
> x
[1] 1.20 2.30 4.21 0.00 2.10 3.40
> sort(x,decreasing=T,index.return=F)
[1] 4.21 3.40 2.30 2.10 1.20 0.00

> x
[1] 1.20 2.30 4.21 0.00 2.10 3.40
> res<-sort(x,decreasing=T,index.return=T)
> res$x
[1] 4.21 3.40 2.30 2.10 1.20 0.00
> res$ix
[1] 3 6 2 5 1 4

> x
[1] 1.20 2.30 4.21 0.00 2.10 3.40
> sort(x,decreasing=F,index.return=F)
[1] 0.00 1.20 2.10 2.30 3.40 4.21

> x
[1] 1.20 2.30 4.21 0.00 2.10 3.40
> res<-sort(x,decreasing=F,index.return=T)
> res$x
[1] 0.00 1.20 2.10 2.30 3.40 4.21
> res$ix
[1] 4 1 5 2 6 3
```

- **Osservazioni:** Equivale alla funzione `order()` quando `index.return = T`.

rev()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** elementi di un vettore in ordine invertito
- **Formula:**

$$x_n,\, x_{n-1},\, \ldots,\, x_1$$

- **Esempio:**

```
> x
[1] 1.20 2.30 4.21 0.00 2.10 3.40
> rev(x)
[1] 3.40 2.10 0.00 4.21 2.30 1.20

> x
[1]  1.2  4.2  4.5 -5.6  6.5  1.2
> rev(x)
[1]  1.2  6.5 -5.6  4.5  4.2  1.2
```

order()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

`decreasing = T / F` decremento oppure incremento

- **Significato:** restituisce la posizione di ogni elemento di x se questo fosse ordinato in maniera decrescente oppure crescente
- **Esempio:**

```
> x
[1] 1.20 2.30 4.21 0.00 2.10 3.40
> order(x,decreasing=F)
[1] 4 1 5 2 6 3

> x
[1] 1.20 2.30 4.21 0.00 2.10 3.40
> order(x,decreasing=T)
[1] 3 6 2 5 1 4

> x
[1]  1.6  6.8  7.7  7.2  5.4  7.9  8.0  8.0  3.4 12.0
> sort(x,decreasing=F)
[1]  1.6  3.4  5.4  6.8  7.2  7.7  7.9  8.0  8.0 12.0
> x[order(x,decreasing=F)]
[1]  1.6  3.4  5.4  6.8  7.2  7.7  7.9  8.0  8.0 12.0
```

rank()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `ties.method = average / first / random / max / min` metodo da utilizzare in presenza di ties

- **Significato:** rango di x ossia viene associato ad ogni elemento del vettore x il posto occupato nello stesso vettore ordinato in modo crescente
- **Esempio:**

```
> x
[1] 1 2 3 4 2 3 4
> rank(x,ties.method="average")
[1] 1.0 2.5 4.5 6.5 2.5 4.5 6.5

> x
[1]  1  2  3  4  5  6  7  8  9 10
> rank(x,ties.method="average")
[1]  1  2  3  4  5  6  7  8  9 10

> x
[1] 10  9  8  7  6  5  4  3  2  1
> rank(x,ties.method="average")
[1] 10  9  8  7  6  5  4  3  2  1
```

1.12 Funzioni di arrotondamento

trunc()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** tronca la parte decimale
- **Formula:**

$$[x]$$

- **Esempio:**

```
> trunc(x=2)
[1] 2

> trunc(x=2.999)
[1] 2

> trunc(x=-2.01)
[1] -2
```

floor()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** arrotonda all'intero inferiore
- **Formula:**

$$\lfloor x \rfloor = \begin{cases} x & \text{se } x \text{ è intero} \\ [x] & \text{se } x \text{ è positivo non intero} \\ [x]-1 & \text{se } x \text{ è negativo non intero} \end{cases}$$

- **Esempio:**

```
> floor(x=2)
[1] 2

> floor(x=2.99)
[1] 2

> floor(x=-2.01)
[1] -3
```

ceiling()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** arrotonda all'intero superiore
- **Formula:**

$$\lceil x \rceil = \begin{cases} x & \text{se } x \text{ è intero} \\ [x]+1 & \text{se } x \text{ è positivo non intero} \\ [x] & \text{se } x \text{ è negativo non intero} \end{cases}$$

- **Esempio:**

```
> ceiling(x=2)
[1] 2

> ceiling(x=2.001)
[1] 3

> ceiling(x=-2.01)
[1] -2
```

round()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

  `digits` valore naturale n

- **Significato:** arrotonda al numero di cifre specificato da n
- **Esempio:**

```
> pi
[1] 3.141593
> round(x=pi,digits=4)
[1] 3.1416

> exp(1)
[1] 2.718282
> round(x=exp(1),digits=3)
[1] 2.718
```

signif()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

  `digits` valore naturale n

- **Significato:** arrotonda al numero di cifre significative specificate da n
- **Esempio:**

```
> pi
[1] 3.141593
> signif(x=pi,digits=4)
[1] 3.142

> exp(1)
[1] 2.718282
> signif(x=exp(1),digits=3)
[1] 2.72
```

fractions()

- **Package:** `MASS`
- **Parametri:**

  `x` oggetto numerico

- **Significato:** trasforma un valore decimale in frazionario
- **Esempio:**

```
> fractions(x=2.3)
[1] 23/10

> fractions(x=1.34)
[1] 67/50

> x<-matrix(data=c(1.2,34,4.3,4.2),nrow=2,ncol=2,byrow=F)
> x
     [,1] [,2]
[1,]  1.2  4.3
[2,] 34.0  4.2
> fractions(x)
     [,1]  [,2]
[1,]   6/5 43/10
[2,]    34  21/5
```

rational()

- **Package:** `MASS`
- **Parametri:**

  `x` oggetto numerico

- **Significato:** approssimazione razionale
- **Esempio:**

```
> mat<-matrix(data=c(1.2,34,4.3,4.2),nrow=2,ncol=2,byrow=F)
> mat
     [,1] [,2]
[1,]  1.2  4.3
[2,] 34.0  4.2
> det(mat)
[1] -141.16
> # matrice invertibile
> solve(mat)%*%mat
              [,1]          [,2]
[1,] 1.000000e+00 -2.303930e-17
[2,] 2.428613e-17  1.000000e+00
> rational(x=solve(mat)%*%mat)
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

1.13 Funzioni avanzate

gamma()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $x > 0$

- **Significato:** funzione gamma
- **Formula:**

$$\Gamma(x) = \int_0^{+\infty} u^{x-1}\, e^{-u}\, du$$

- **Esempio:**

```
> gamma(x=3.45)
[1] 3.146312

> gamma(x=5)
[1] 24
```

lgamma()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $x > 0$

- **Significato:** logaritmo naturale della funzione gamma
- **Formula:**

$$\log\big(\Gamma(x)\big)$$

- **Esempio:**

```
> log(gamma(x=3.45))
[1] 1.146231
> lgamma(x=3.45)
[1] 1.146231

> log(gamma(x=5))
[1] 3.178054
> lgamma(x=5)
[1] 3.178054
```

digamma()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $x > 0$

- **Significato:** funzione digamma
- **Formula:**

$$\Psi(x) = \frac{d}{dx} \log\big(\Gamma(x)\big)$$

- **Esempio:**

```
> digamma(x=2.45)
[1] 0.6783387

> digamma(x=5.3)
[1] 1.570411
```

trigamma()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $x > 0$

- **Significato:** derivata prima della funzione digamma
- **Formula:**

$$\frac{d}{dx}\,\Psi(x)$$

- **Esempio:**

```
> trigamma(x=2.45)
[1] 0.5024545

> trigamma(x=5.3)
[1] 0.2075909
```

psigamma()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico tale che $x > 0$

  `deriv` valore naturale n

- **Significato:** derivata n-esima della funzione digamma
- **Formula:**

$$\frac{d^n}{dx}\,\Psi(x)$$

- **Esempio:**

```
> psigamma(x=2.45,deriv=0)  # comando digamma
[1] 0.6783387
> digamma(x=2.45)
[1] 0.6783387

> psigamma(x=5.3,deriv=1)  # comando trigamma
[1] 0.2075909
> trigamma(x=5.3)
[1] 0.2075909
```

beta()

- **Package:** `base`
- **Parametri:**
  - `a` valore numerico tale che $a > 0$
  - `b` valore numerico tale che $b > 0$
- **Significato:** funzione beta
- **Formula:**

$$B(a,\, b) \,=\, \frac{\Gamma(a)\,\Gamma(b)}{\Gamma(a+b)} \,=\, \int_0^1 u^{a-1}\,(1-u)^{b-1}\,du$$

- **Esempio:**

```
> a<-3.45
> b<-2.3
> gamma(a)*gamma(b)/gamma(a+b)
[1] 0.04659344
> beta(a=3.45,b=2.3)
[1] 0.04659344

> a<-5
> b<-4
> gamma(a)*gamma(b)/gamma(a+b)
[1] 0.003571429
> beta(a=5,b=4)
[1] 0.003571429
```

lbeta()

- **Package:** `base`
- **Parametri:**
  - `a` valore numerico tale che $a > 0$
  - `b` valore numerico tale che $b > 0$
- **Significato:** logaritmo naturale della funzione beta
- **Formula:**

$$\log\left(B(a,\, b)\right)$$

- **Esempio:**

```
> a<-3.45
> b<-2.3
> log(gamma(a)*gamma(b)/gamma(a+b))
[1] -3.066296
> lbeta(a=3.45,b=2.3)
[1] -3.066296

> a<-5
> b<-4
> log(gamma(a)*gamma(b)/gamma(a+b))
[1] -5.63479
> lbeta(a=5,b=4)
[1] -5.63479
```

sigmoid()

- **Package:** `e1071`
- **Parametri:**

  `x` valore numerico

- **Significato:** funzione sigmoide
- **Formula:**

$$S(x) = (1 + e^{-x})^{-1}$$

- **Esempio:**

```
> x<-3.45
> (1+exp(-x))^(-1)
[1] 0.9692311
> sigmoid(x=3.45)
[1] 0.9692311

> x<--1.7
> (1+exp(-x))^(-1)
[1] 0.1544653
> sigmoid(x=-1.7)
[1] 0.1544653
```

dsigmoid()

- **Package:** `e1071`
- **Parametri:**

  `x` valore numerico

- **Significato:** derivata prima della funzione sigmoide
- **Formula:**

$$\frac{d}{dx} S(x) = \frac{e^x}{(1 + e^x)^2}$$

- **Esempio:**

```
> x<-3.45
> exp(x)/(1+exp(x))**2
[1] 0.02982214
> dsigmoid(x=3.45)
[1] 0.02982214

> x<--1.7
> exp(x)/(1+exp(x))**2
[1] 0.1306057
> dsigmoid(x=-1.7)
[1] 0.1306057
```

d2sigmoid()

- **Package:** `e1071`
- **Parametri:**

  `x` valore numerico

- **Significato:** derivata seconda della funzione sigmoide

- **Formula:**

$$\frac{d^2}{dx}\,S(x) \;=\; \frac{e^x\,(1-e^x)}{(1+e^x)^3}$$

- **Esempio:**

```
> x<-3.45
> (exp(x)*(1-exp(x)))/(1+exp(x))**3
[1] -0.02798695
> d2sigmoid(x=3.45)
[1] -0.02798695

> x<--1.7
> (exp(x)*(1-exp(x)))/(1+exp(x))**3
[1] 0.09025764
> d2sigmoid(x=-1.7)
[1] 0.09025764
```

1.14 Funzioni sui numeri complessi

complex()

- **Package:** `base`
- **Parametri:**

  `real` parte reale α

  `imaginary` parte immaginaria β

  `modulus` modulo r

  `argument` argomento ϕ

- **Significato:** numero complesso
- **Formula:**

$$\begin{aligned}
\alpha + i\,\beta &= r\,(\cos(\phi) + i\,\sin(\phi)) \\
\alpha &= r\,\cos(\phi) \\
\beta &= r\,\sin(\phi) \\
r &= \sqrt{\alpha^2+\beta^2} \\
\phi &= \arctan\left(\frac{\beta}{\alpha}\right)
\end{aligned}$$

- **Esempio:**

```
> complex(real=1,imaginary=3)                          # coordinate cartesiane
[1] 1+3i

> complex(modulus=Mod(1+3i),argument=Arg(1+3i))        # coordinate polari
[1] 1+3i

> complex(real=-3,imaginary=4)                         # coordinate cartesiane
[1] -3+4i

> complex(modulus=Mod(-3+4i),argument=Arg(-3+4i))      # coordinate polari
[1] -3+4i
```

Re()

- **Package:** `base`
- **Parametri:**

  `x` numero complesso

- **Significato:** parte reale
- **Formula:**

$$\alpha$$

- **Esempio:**

```
> Re(x=2+3i)
[1] 2

> Re(x=-3+4i)
[1] -3
```

Im()

- **Package:** `base`
- **Parametri:**

  `x` numero complesso

- **Significato:** parte immaginaria
- **Formula:**

$$\beta$$

- **Esempio:**

```
> Im(x=-2+3i)
[1] 3

> Im(x=-3+4i)
[1] 4
```

Mod()

- **Package:** `base`
- **Parametri:**

  `x` numero complesso

- **Significato:** modulo
- **Formula:**

$$r = \sqrt{\alpha^2 + \beta^2}$$

- **Esempio:**

```
> x
[1] 2+3i
> sqrt(2**2+3**2)
[1] 3.605551
> Mod(x=2+3i)
[1] 3.605551
```

```
> x
[1] -3+4i
> sqrt((-3)**2+4**2)
[1] 5
> Mod(x=-3+4i)
[1] 5
```

- **Osservazioni:** Equivale alla funzione `abs()`.

Arg()

- **Package:** `base`
- **Parametri:**

  `x` numero complesso

- **Significato:** argomento
- **Formula:**

$$\phi = \arctan\left(\frac{\beta}{\alpha}\right)$$

- **Esempio:**

```
> x
[1] 2+3i
> atan(3/2)
[1] 0.9827937
> Arg(x=2+3i)
[1] 0.9827937
```

```
> x
[1] 4+5i
> atan(5/4)
[1] 0.8960554
> Arg(x=4+5i)
[1] 0.8960554
```

Conj()

- **Package:** `base`
- **Parametri:**

  `x` numero complesso

- **Significato:** coniugato
- **Formula:**

$$\alpha - i\,\beta$$

- **Esempio:**

```
> Conj(x=2+3i)
[1] 2-3i
```

```
> Conj(x=-3+4i)
[1] -3-4i
```

is.real()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** segnalazione di valore numerico reale
- **Esempio:**

```
> is.real(x=2+3i)
[1] FALSE

> is.real(x=4)
[1] TRUE
```

is.complex()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** segnalazione di valore numerico complesso
- **Esempio:**

```
> is.complex(x=2+3i)
[1] TRUE
>
> is.complex(x=4)
[1] FALSE
```

1.15 Funzioni cumulate

cumsum()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** somma cumulata
- **Formula:**

$$\sum_{j=1}^{i} x_j \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> x
[1] 1 2 4 3 5 6
> cumsum(x)
[1]  1  3  7 10 15 21

> x
[1] 1.0 2.3 4.5 6.7 2.1
> cumsum(x)
[1]  1.0  3.3  7.8 14.5 16.6
```

cumprod()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** prodotto cumulato
- **Formula:**

$$\prod_{j=1}^{i} x_j \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> x
[1] 1 2 4 3 5 6
> cumprod(x)
[1]   1   2   8  24 120 720

> x
[1] 1.0 2.3 4.5 6.7 2.1
> cumprod(x)
[1]   1.0000   2.3000  10.3500  69.3450 145.6245
```

cummin()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** minimo cumulato
- **Formula:**

$$\min(x_1, x_2, \ldots, x_i) \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> x
[1] 3 4 3 2 4 1
> cummin(x)
[1] 3 3 3 2 2 1

> x
[1] 1 3 2 4 5 1
> cummin(x)
[1] 1 1 1 1 1 1
```

cummax()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** massimo cumulato
- **Formula:**

$$\max(x_1, x_2, \ldots, x_i) \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> x
[1] 1 3 2 4 5 1
> cummax(x)
[1] 1 3 3 4 5 5

> x
[1] 1 3 2 4 5 1
> cummax(x)
[1] 1 3 3 4 5 5
```

1.16 Funzioni in parallelo

pmin()

- **Package:** `base`
- **Parametri:**
  - `x` vettore numerico di dimensione n
  - `y` vettore numerico di dimensione n
- **Significato:** minimo in parallelo
- **Formula:**

$$\min(x_i, y_i) \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> x
[1] 1.20 2.30 0.11 4.50
> y
[1] 1.1 2.1 1.3 4.4
> pmin(x,y)
[1] 1.10 2.10 0.11 4.40

> x
[1] 1.20 2.30 0.11 4.50
> y
[1] 1.1 2.1 1.1 2.1
> pmin(x,y)
[1] 1.1 2.1 0.11 2.1
```

pmax()

- **Package:** `base`
- **Parametri:**
  - `x` vettore numerico di dimensione n
  - `y` vettore numerico di dimensione n
- **Significato:** massimo in parallelo
- **Formula:**

$$\max(x_i, y_i) \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> x
[1] 1.20 2.30 0.11 4.50
> y
[1] 1.1 2.1 1.3 4.4
> pmax(x,y)
[1] 1.2 2.3 1.3 4.5

> x
[1] 1.20 2.30 0.11 4.50
> y
[1] 1.1 2.1 1.1 2.1
> pmax(x,y)
[1] 1.2 2.3 1.1 4.5
```

1.17 Funzioni di analisi numerica

optimize()

- **Package:** `stats`
- **Parametri:**

  `f` funzione $f(x)$
  `lower` estremo inferiore
  `upper` estremo superiore
  `maximum = T / F` massimo oppure minimo
  `tol` tolleranza

- **Significato:** ricerca di un massimo oppure di un minimo
- **Output:**

  `minimum` punto di minimo
  `maximum` punto di massimo
  `objective` valore assunto dalla funzione nel punto individuato

- **Formula:**

`maximum = T`

$$\max_x f(x)$$

`maximum = F`

$$\min_x f(x)$$

- **Esempio:**

```
> f<-function(x) x*exp(-x**3)-(log(x))**2
> optimize(f,lower=0.3,upper=1.5,maximum=T,tol=1e-4)
$maximum
[1] 0.8374697

$objective
[1] 0.4339975

> f<-function(x) (x-0.1)^2
> optimize(f,lower=0,upper=1,maximum=F,tol=1e-4)
$minimum
[1] 0.1

$objective
[1] 7.70372e-34
```

uniroot()

- **Package:** `stats`
- **Parametri:**

  `f` funzione $f(x)$

  `lower` estremo inferiore

  `upper` estremo superiore

  `tol` tolleranza

  `maxiter` mumero massimo di iterazioni

- **Significato:** ricerca di uno zero
- **Output:**

  `root` radice

  `f.root` valore assunto dalla funzione nel punto individuato

  `iter` numero di iterazioni

  `estim.prec` tolleranza

- **Formula:**

$$f(x) = 0$$

- **Esempio:**

```
> f<-function(x) exp(-x)-x
> uniroot(f,lower=0,upper=1,tol=1e-4,maxiter=1000)
```

polyroot()

- **Package:** `base`
- **Parametri:**

  `a` vettore dei k coefficienti di un polinomio di ordine $k-1$

- **Significato:** ricerca di uno zero in un polinomio
- **Formula:**

$$a_1 + a_2\,x + a_3\,x^2 + \cdots + a_k\,x^{k-1} = 0$$

- **Esempio:**

```
> k<-3
> # funzione polinomiale di secondo grado
> a1<-3
> a2<--2
> a3<-2
> a<-c(a1,a2,a3)
> polyroot(a)
[1] 0.5+1.118034i 0.5-1.118034i
> # verifica radici
> radice1<-0.5+1.1180340i
> a1+a2*radice1+a3*radice1**2
[1] -5.0312e-08+0i
> # verifica OK
> radice2<-0.5-1.1180340i
> a1+a2*radice2+a3*radice2**2
[1] -5.0312e-08+0i
> # verifica OK

> k<-4
```

```
> # funzione polinomiale di terzo grado
> a1<-3
> a2<--2
> a3<-2
> a4<--1
> a<-c(a1,a2,a3,a4)
> polyroot(a)
[1] 0.09473214+1.283742i 0.09473214-1.283742i 1.81053571+0.000000i
> # verifica radici
> radice1<-0.09473214+1.283742i
> a1+a2*radice1+a3*radice1**2+a4*radice1**3
[1] 7.477461e-07-5.808714e-07i
> # verifica OK
> radice2<-0.09473214-1.283742i
> a1+a2*radice2+a3*radice2**2+a4*radice2**3
[1] 7.477461e-07+5.808714e-07i
> # verifica OK
> radice3<-1.81053571+0.000000i
> a1+a2*radice3+a3*radice3**2+a4*radice3**3
[1] 1.729401e-08+0i
> # verifica OK
```

D()

- **Package:** `stats`
- **Parametri:**

  `expr` espressione contenente la funzione $f(x)$ da derivare

  `name` variabile x di derivazione

- **Significato:** derivata simbolica al primo ordine
- **Formula:**

$$\frac{d}{dx} f(x)$$

- **Esempio:**

```
> D(expr=expression(exp(-x)-x),name="x")
-(exp(-x) + 1)

> D(expr=expression(x*exp(-a)),name="x")
exp(-a)
```

integrate()

- **Package:** `stats`
- **Parametri:**

  `f` funzione $f(x)$

  `lower` estremo inferiore a di integrazione

  `upper` estremo superiore b di integrazione

  `subdivisions` mumero di suddivisioni dell'intervallo di integrazione

- **Significato:** integrazione numerica
- **Output:**

  `value` integrale definito

- **Formula:**

$$\int_a^b f(x)\,dx$$

- **Esempio:**

```
> f<-function(x) exp(-x)
> integrate(f,lower=1.2,upper=2.3,subdivisions=150)
0.2009354 with absolute error < 2.2e-15

> f<-function(x) sqrt(x)
> integrate(f,lower=2.1,upper=4.5,subdivisions=150)
4.335168 with absolute error < 4.8e-14
```

1.18 Costanti

pi

- **Package:** base
- **Significato:** pi greco
- **Formula:**

$$\pi$$

- **Esempio:**

```
> pi
[1] 3.141593

> 2*pi
[1] 6.283185
```

Inf

- **Package:** base
- **Significato:** infinito
- **Formula:**

$$\infty$$

- **Esempio:**

```
> 2/0
[1] Inf

> -2/0
[1] -Inf
```

NaN

- **Package:** base
- **Significato:** not a number
- **Esempio:**

```
> Inf-Inf
[1] NaN

> 0/0
[1] NaN
```

NA

- **Package:** `base`
- **Significato:** not available
- **Esempio:**

```
> x<-c(1.2,3.4,5.6,NA)
> mean(x)
[1] NA
> mean(x,na.rm=T)
[1] 3.4
```

NULL

- **Package:** `base`
- **Significato:** oggetto nullo
- **Esempio:**

```
> x
  a   b   c
1.2 3.4 5.6
> names(x)<-NULL
> x
[1] 1.2 3.4 5.6
```

TRUE

- **Package:** `base`
- **Significato:** vero
- **Esempio:**

```
> T
[1] TRUE

> TRUE & TRUE
[1] TRUE
```

T

- **Package:** `base`
- **Significato:** vero
- **Esempio:**

```
> T
[1] TRUE

> T & T
[1] TRUE
```

FALSE

- **Package:** `base`
- **Significato:** falso
- **Esempio:**

```
> FALSE
[1] FALSE

> FALSE | TRUE
[1] TRUE
```

F

- **Package:** `base`
- **Significato:** falso
- **Esempio:**

```
> F
[1] FALSE

> F | T
[1] TRUE
```

1.19 Miscellaneous

list()

- **Package:** `base`
- **Significato:** creazione di un oggetto lista
- **Esempio:**

```
> x
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1
> y
[1] 4.5 5.4 6.1 6.1 5.4
> lista<-list(x=x,y=y)
> lista
$x
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1
$y
[1] 4.5 5.4 6.1 6.1 5.4
> lista[1]
$x
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1
> lista$x
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1
> lista[[1]]
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1
> lista[[1]][1]
[1] 7.8
> lista[2]
$y
[1] 4.5 5.4 6.1 6.1 5.4
> lista$y
[1] 4.5 5.4 6.1 6.1 5.4
```

```
> lista[[2]]
[1] 4.5 5.4 6.1 6.1 5.4
> lista[[2]][1]
[1] 4.5

> x
[1] 1.0 2.3 4.5 6.7 8.9
> y
[1] 154 109 137 115 140
> z
[1] 108 115 126  92 146
> lista<-list(x=x,y=y,z=z)
> lista
$x
[1] 1.0 2.3 4.5 6.7 8.9
$y
[1] 154 109 137 115 140
$z
[1] 108 115 126  92 14
> lista[1]
$x
[1] 1.0 2.3 4.5 6.7 8.9
> lista$x
[1] 1.0 2.3 4.5 6.7 8.9
> lista[[1]]
[1] 1.0 2.3 4.5 6.7 8.9
> lista[[1]][1]
[1] 1
> lista[2]
$y
[1] 154 109 137 115 140
> lista$y
[1] 154 109 137 115 140
> lista[[2]]
[1] 154 109 137 115 140
> lista[[2]][1]
[1] 154
> lista[3]
$z
[1] 108 115 126  92 146
> lista$z
[1] 108 115 126  92 146
> lista[[3]]
[1] 108 115 126  92 146
> lista[[3]][1]
[1] 108

> x
[1] 1 2 3
> y
[1] 11 12 13 14 15
> lista<-list(x,y)
> lista
[[1]]
[1] 1 2 3
[[2]]
[1] 11 12 13 14 15
> names(lista)
NULL

> x
[1] 1 2 3
> y
```

```
[1] 11 12 13 14 15
> lista<-list(A=x,B=y)
> lista
$A
[1] 1 2 3
$B
[1] 11 12 13 14 15
> names(lista)
[1] "A" "B"
```

lapply()

- **Package:** `base`
- **Parametri:**

  `x` oggetto lista

  `FUN` funzione

- **Significato:** applica la funzione `FUN` ad ogni elemento di lista
- **Esempio:**

```
> vec1
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1
> mean(vec1)
[1] 7.0125
> vec2
[1] 4.5 5.4 6.1 6.1 5.4
> mean(vec2)
[1] 5.5
> x<-list(vec1=vec1,vec2=vec2)
> lapply(x,FUN=mean)
$vec1
[1] 7.0125
$vec2
[1] 5.5
```

```
> vec1
[1] 1.0 2.3 4.5 6.7 8.9
> sd(vec1)
[1] 3.206556
> vec2
[1] 154 109 137 115 140
> sd(vec2)
[1] 18.61451
> vec3
[1] 108 115 126  92 146
> sd(vec3)
[1] 20.19406
> x<-list(vec1=vec1,vec2=vec2,vec3=vec3)
> lapply(x,FUN=sd)
$vec1
[1] 3.206556
$vec2
[1] 18.61451
$vec3
[1] 20.19406
```

duplicated()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** segnalazione di valori duplicati
- **Esempio:**

```
> x
[1] 1 2 1 3 2 2 4
> duplicated(x)
[1] FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE

> x
[1] 1 2 1 2 1 2
> duplicated(x)
[1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE

> x
[1] 12 -3  7 12  4 -3 12  7 -3
> unique(x[duplicated(x)])  # valori duplicati di x
[1] 12 -3  7
```

.Last.value

- **Package:** `base`
- **Significato:** ultimo valore calcolato
- **Esempio:**

```
> 2+4
[1] 6
> .Last.value
[1] 6

> 3*4**4.2
[1] 1013.382
> .Last.value
[1] 1013.382
```

identical

- **Package:** `base`
- **Significato:** uguaglianza tra due oggetti
- **Esempio:**

```
> u
[1] 1 2 3
> v
[1] 1 2 4
> {if(identical(u,v)) print("uguali") else print("non uguali")}
[1] "non uguali"

> u
[1] 1 2 3
> v
```

```
[1] 1 3 2
> identical(u,v)
[1] FALSE
```

any()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** restituisce `TRUE` se almeno un elemento del vettore soddisfa ad una condizione fissata
- **Esempio:**

```
> x
[1] 3 4 3 2 4 1
> x<2
[1] FALSE FALSE FALSE FALSE FALSE  TRUE
> any(x<2)
[1] TRUE

> x
[1] 1 2 3 4 5 6 7 8
> x>4
[1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
> any(x>4)
[1] TRUE
```

all()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** restituisce `TRUE` se tutti gli elementi del vettore soddisfano ad una condizione fissata
- **Esempio:**

```
> x
[1] 3 4 3 2 4 1
> x<2
[1] FALSE FALSE FALSE FALSE FALSE  TRUE
> all(x<2)
[1] FALSE

> x
[1] 1 2 3 4 5 6 7 8
> x>4
[1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE
> all(x>4)
[1] FALSE
```

match()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `table` vettore numerico y di dimensione m

  `nomatch` alternativa da inserire al posto di `NA`

- **Significato:** per ogni elemento di x restituisce la posizione della prima occorrenza in y
- **Esempio:**

```
> x
[1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
> match(x,table=c(2,4),nomatch=0)
[1] 0 0 0 1 1 1 0 0 0 2 2 2 0 0 0

> x
[1] 1 1 1 2 2 2 3 3 3 4 4 4 5 5 5
> match(x,table=c(2,4),nomatch=NA)
[1] NA NA NA  1  1  1 NA NA NA  2  2  2 NA NA NA

> match(x=c(-3,3),table=c(5,33,3,6,-3,-4,3,5,-3),nomatch=NA)
[1] 5 3
```

outer()

- **Package:** `base`
- **Parametri:**

  `X` vettore numerico x di dimensione n

  `Y` vettore numerico y di dimensione m

  `FUN` funzione $f(x, y)$

- **Significato:** applica la funzione `FUN` ad ogni coppia ordinata costituita da un elemento di x ed uno di y
- **Formula:**

$$f(x_i, y_j) \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, m$$

- **Esempio:**

```
> outer(X=c(1,2,2,4),Y=c(1.2,2.3),FUN="+")
     [,1] [,2]
[1,]  2.2  3.3
[2,]  3.2  4.3
[3,]  3.2  4.3
[4,]  5.2  6.3

> outer(X=c(1,2,2,4),Y=c(1.2,2.3),FUN="*")
     [,1] [,2]
[1,]  1.2  2.3
[2,]  2.4  4.6
[3,]  2.4  4.6
[4,]  4.8  9.2
```

expression()

- **Package:** `base`
- **Parametri:**

  `x` oggetto

- **Significato:** crea una espressione simbolica
- **Esempio:**

```
> u
[1] 4.3 5.5 6.8 8.0
> w
[1] 4 5 6 7
> z<-expression(x=u/w)

> u
[1] 1.2 3.4 4.5
> w
[1]  1  2 44
> z<-expression(x=u*w)
```

eval()

- **Package:** `base`
- **Parametri:**

  `expr` espressione simbolica

- **Significato:** valuta una espressione simbolica
- **Esempio:**

```
> u
[1] 4.3 5.5 6.8 8.0
> w
[1] 4 5 6 7
> z<-expression(x=u/w)
> eval(expr=z)
[1] 1.075000 1.100000 1.133333 1.142857

> u
[1] 1.2 3.4 4.5
> w
[1]  1  2 44
> z<-expression(expr=u*w)
> eval(z)
[1]   1.2   6.8 198.0
```

replace()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `list` indice dell'elemento da rimpiazzare

  `values` valore da inserire

- **Significato:** rimpiazza un elemento del vettore x

- **Esempio:**

```
> x
[1] 1 2 3 4 5 6 7 8
> replace(x,list=1,values=10)
[1] 10  2  3  4  5  6  7  8

> x
[1] 1 2 3 4 5 6 7 8
> replace(x,list=1,values=10)
[1] 10  2  3  4  5  6  7  8
```

- **Osservazioni:** Il vettore x rimane invariato.

e

- **Package:** `base`
- **Significato:** scrittura veloce di un valore numerico potenza di 10
- **Esempio:**

```
> 1e3
[1] 1000

> -2e-2
[1] -0.02

> 1e-2
[1] 0.01

> 3e4
[1] 30000
```

even()

- **Package:** `gtools`
- **Parametri:**

  `x` valore naturale

- **Significato:** verifica numero pari
- **Esempio:**

```
> even(x=22)
[1] TRUE

> even(x=7)
[1] FALSE
```

odd()

- **Package:** `gtools`
- **Parametri:**

  `x` valore naturale

- **Significato:** verifica numero dispari

- **Esempio:**

```
> odd(x=22)
[1] FALSE

> odd(x=7)
[1] TRUE
```

'

- **Package:** `base`
- **Significato:** notazione polacca inversa (RPN)
- **Esempio:**

```
> '+'(1,2)
[1] 3

> '*'(3,4.2)
[1] 12.6
```

- **Osservazioni:** RPN = Reverse Polish Notation.

Capitolo 2

Vettori, Matrici ed Array

2.1 Creazione di Vettori

c()

- **Package:** `base`
- **Parametri:**

  `recursive = T / F` concatenazione per oggetti di tipo `list()`

- **Significato:** funzione di concatenazione
- **Esempio:**

```
> x<-c(1.2,3.4,5.6,7.8)
> x
[1] 1.2 3.4 5.6 7.8
> x<-c(x,9.9)
> x
[1] 1.2 3.4 5.6 7.8 9.9

> x<-c(1.2,3.4,5.6,7.8)
> x
[1] 1.2 3.4 5.6 7.8
> x[5]<-9.9
> x
[1] 1.2 3.4 5.6 7.8 9.9

> x<-c("a","b")
> x
[1] "a" "b"

> x<-c('a','b')
> x
[1] "a" "b"

> x<-c("a","b","a","a","b")
> x
[1] "a" "b" "a" "a" "b"
> x<-c(x,"a")
> x
[1] "a" "b" "a" "a" "b" "a"

> x<-c("a","b","a","a","b")
> x
[1] "a" "b" "a" "a" "b"
> x[6]<-"a"
> x
[1] "a" "b" "a" "a" "b" "a"
```

```
> x<-c("a",1)
> x
[1] "a" "1"
> # valori numerici trasformati in carattere
> x<-c(x,2)
> x
[1] "a" "1" "2"

> lista<-list(primo=c(1,2,3),secondo=c(1.2,5.6))
> vettore<-c(lista,recursive=T)
> vettore
  primo1   primo2   primo3 secondo1 secondo2
     1.0      2.0      3.0      1.2      5.6
```

- **Osservazioni 1:** Se il vettore è molto lungo, conviene utilizzare la funzione `scan()`.
- **Osservazioni 2:** I vettori alfanumerici possono essere definiti usando " oppure '.

scan()

- **Package:** `base`
- **Significato:** creazione di un vettore
- **Esempio:**

```
> # creazione di un vettore numerico
> x<-scan()
1: 1.2
2: 3.4
3: 0.45
4:
Read 3 items
> x
[1] 1.20 3.40 0.45

> x<-scan()
1: 1.2 3.4 0.45
4:
Read 3 items
> x
[1] 1.20 3.40 0.45

> # creazione di un vettore di caratteri
> x<-scan(what="character")
1: a
2: b
3: a
4:
Read 3 items
> x
[1] "a" "b" "a"

> x<-scan(what="character")
1: a b a
4:
Read 3 items
> x
[1] "a" "b" "a"
```

[ ]

- **Package:** `base`
- **Parametri:**

  `x` vettore alfanumerico di dimensione n

- **Significato:** estrazione di elementi da un vettore
- **Esempio:**

```
> x
[1] 1.2 3.4 5.6 7.8 9.0 9.9
> # elemento di posto 2
> x[2]
[1] 3.4
> # elementi di posto 1,3,4
> x[c(1,3,4)]
[1] 1.2 5.6 7.8
> # elementi di posto 1,2,3
> x[1:3]
[1] 1.2 3.4 5.6
> # tutti gli elementi eccetto quelli di posto 1,2,3
> x[-c(1:3)]
[1] 7.8 9.0 9.9
> x[-(1:3)]
[1] 7.8 9.0 9.9
> # gli elementi in un set dato
> x[x %in% c(1.2,7.8)]
[1] 1.2 7.8
> # elementi maggiori di 6.3
> x[x>6.3]
[1] 7.8 9.0 9.9
> # elementi maggiori di 6.3 e minori di 9.7
> x[x>6.3 & x<9.7]
[1] 7.8 9.0
> # elementi il cui indice corrisponde a T
> x[c(T,T,F,F,T,T)]
[1] 1.2 3.4 9.0 9.9
> # non posso richiamare elementi il cui
> # indice supera la lunghezza del vettore
> x[7]
[1] NA
> # indice nullo significa non
> # considerare nessun elemento
> x[0]
numeric(0)
> # indice NA significa
> # restituire NA
> x[c(1,2,NA)]
[1] 1.2 3.4  NA
> # definizione di etichette
> names(x)<-c("a","b","c","d","e","f")
> x
  a   b   c   d   e   f
1.2 3.4 5.6 7.8 9.0 9.9
> # elemento con etichetta "a"
> x["a"]
  a
1.2
```

names()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** assegnazioni di nomi agli elementi di un vettore
- **Esempio:**

```
> x
[1] 1.2 3.4 5.6
> names(x)
NULL
> names(x)<-c("primo","secondo","terzo")
> x
  primo secondo   terzo
    1.2     3.4     5.6
> names(x)
[1] "primo"   "secondo" "terzo"
> x[c("primo","terzo")]
primo terzo
  1.2   5.6
```

vector()

- **Package:** `base`
- **Parametri:**

  `mode = numeric / complex / logical` tipo di oggetto

  `length` valore n della dimensione

- **Significato:** inizializzazione di un vettore di dimensione n
- **Esempio:**

```
> x<-vector(mode="numeric",length=5)
> x
[1] 0 0 0 0 0

> x<-vector(mode="complex",length=3)
> x
[1] 0+0i 0+0i 0+0i

> x<-vector(mode="logical",length=4)
> x
[1] FALSE FALSE FALSE FALSE
```

numeric()

- **Package:** `base`
- **Parametri:**

  `length` dimensione

- **Significato:** inizializzazione di un vettore numerico di dimensione n
- **Esempio:**

```
> x<-numeric(length=5)
> x
[1] 0 0 0 0 0

> x<-numeric(length=4)
> x
[1] 0 0 0 0
```

complex()

- **Package:** `base`
- **Parametri:**

  `length` dimensione

- **Significato:** inizializzazione di un vettore complesso di dimensione n
- **Esempio:**

```
> x<-complex(length=5)
> x
[1] 0+0i 0+0i 0+0i 0+0i 0+0i

> x<-complex(length=4)
> x
[1] 0+0i 0+0i 0+0i 0+0i
```

logical()

- **Package:** `base`
- **Parametri:**

  `length` dimensione

- **Significato:** inizializzazione di un vettore logico di dimensione n
- **Esempio:**

```
> x<-logical(length=5)
> x
[1] FALSE FALSE FALSE FALSE FALSE

> x<-logical(length=4)
> x
[1] FALSE FALSE FALSE FALSE
```

head()

- **Package:** `utils`
- **Parametri:**

  `x` vettore numerico di dimensione m

  `n` numero di elementi

- **Significato:** seleziona i primi n elementi
- **Esempio:**

```
> x
[1] 1.2 3.2 3.3 2.5 5.0 5.6
> head(x,n=2)
[1] 1.2 3.2

> x
[1]  4.5  6.7  8.9  7.7 11.2
> m<-length(x)
> m
[1] 5
> head(x,n=3)
[1] 4.5 6.7 8.9
```

tail()

- **Package:** `utils`
- **Parametri:**
  - `x` vettore numerico di dimensione m
  - `n` numero di elementi
- **Significato:** seleziona gli ultimi n elementi
- **Esempio:**

```
> x
[1] 1.2 3.2 3.3 2.5 5.0 5.6
> tail(x,n=3)
[1] 2.5 5.0 5.6

> x
[1]  4.5  6.7  8.9  7.7 11.2
> m<-length(x)
> m
[1] 5
> tail(x,n=3)
[1]  8.9  7.7 11.2
```

%o%

- **Package:** `base`
- **Parametri:**
  - `x` vettore numerico di dimensione n
  - `y` vettore numerico di dimensione m
- **Significato:** prodotto esterno
- **Formula:**

$$x_i\, y_j \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, m$$

- **Esempio:**

```
>x
[1] 1 2 3 4
>n<-length(x)
>n
[1] 4
>y
[1] 1.2 3.4
```

```
>m<-length(y)
>m
[1] 2
>x%o%y
     [,1] [,2]
[1,]  1.2  3.4
[2,]  2.4  6.8
[3,]  3.6 10.2
[4,]  4.8 13.6

>x
[1] 3 4 7
>n<-length(x)
>n
[1] 3
>y
[1] 1.1 2.2 3.3
>m<-length(y)
>m
[1] 3
>x%o%y
     [,1] [,2] [,3]
[1,]  3.3  6.6  9.9
[2,]  4.4  8.8 13.2
[3,]  7.7 15.4 23.1
```

append()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `values` valore v numerico

  `after` valore j naturale

- **Significato:** aggiunge un elemento ad un vettore
- **Formula:**

$$\boxed{\texttt{after} \leq 0}$$

$$v,\, x_1,\, x_2,\, \ldots,\, x_n$$

$$\boxed{\texttt{after} \geq n}$$

$$x_1,\, x_2,\, \ldots,\, x_n,\, v$$

$$\boxed{1 \leq \texttt{after} \leq n-1}$$

$$x_1,\, x_2,\, \ldots,\, x_j,\, v,\, x_{j+1},\, x_{j+2},\, \ldots,\, x_n$$

- **Esempio:**

```
> x
[1] 1.2 3.4 5.6
> append(x,values=6,after=-2)
[1] 6.0 1.2 3.4 5.6

> x
[1] 1.2 3.4 5.6
> append(x,values=6,after=2)
[1] 1.2 3.4 6.0 5.6
```

```
> x
[1] 1.2 3.4 5.6
> append(x,values=6,after=7)
[1] 1.2 3.4 5.6 6.0
```

sapply()

- **Package:** `base`
- **Parametri:**

  `X` vettore numerico di dimensione n

  `FUN` funzione scelta

- **Significato:** applica `FUN` ad ogni elemento del vettore X
- **Esempio:**

```
> sapply(X=c(1.2,3.2,4.5,6.7),FUN=sin)
[1]  0.93203909 -0.05837414 -0.97753012  0.40484992

> sapply(X=c(1.2,3.2,4.5,6.7),FUN=log)
[1] 0.1823216 1.1631508 1.5040774 1.9021075

> a
[1] 2 4 7 3 5 2 9 0
> X
[1] 2 4 6
> confronta<-function(x) which(a>x)
> sapply(X,FUN=confronta)
[[1]]
[1] 2 3 4 5 7

[[2]]
[1] 3 5 7

[[3]]
[1] 3 7
```

2.2 Creazione di Matrici

matrix()

- **Package:** `base`
- **Parametri:**

  `data` vettore numerico di dimensione $n\,m$

  `nrow` numero n di righe

  `ncol` numero m di colonne

  `byrow = T / F` elementi disposti per riga oppure per colonna

  `dimnames` etichette di riga e di colonna

- **Significato:** definizione
- **Esempio:**

```
> n<-2
> m<-3
> x<-c(1,-0.2,3,1.1,-0.3,3.2)
> A<-matrix(x,nrow=n,ncol=m,byrow=T)
> A
     [,1] [,2] [,3]
[1,]  1.0 -0.2  3.0
[2,]  1.1 -0.3  3.2

> n<-3
> m<-2
> x<-c(1,-0.2,3,4,5.6,6.7)
> A<-matrix(x,nrow=n,ncol=m,byrow=F)
> A
     [,1] [,2]
[1,]  1.0  4.0
[2,] -0.2  5.6
[3,]  3.0  6.7

> n<-2
> m<-3
> x<-0
> A<-matrix(x,nrow=n,ncol=m)
> A
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0

> n<-2
> m<-3
> x<-1
> A<-matrix(x,nrow=n,ncol=m)
> A
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1

> n<-3
> m<-3
> x<-1:9
> riga<-c("r1","r2","r3")
> colonna<-c("c1","c2","c3")
> A<-matrix(x,nrow=n,ncol=m,byrow=F,dimnames=list(riga,colonna))
> A
   c1 c2 c3
r1  1  4  7
r2  2  5  8
r3  3  6  9
```

dim()

- **Package:** base
- **Parametri:**

  x vettore numerico di dimensione nm

- **Significato:** dimensione
- **Esempio:**

```
> n<-3
> m<-3
```

```
> x<-1:9
> dim(x)<-c(n,m)
> x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> n<-1
> m<-5
> x<-1:5
> dim(x)<-c(n,m)
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5
```

colnames()

- **Package:** `base`
- **Parametri:**

  `x` matrice di dimensione $n \times m$

- **Significato:** etichette di colonna
- **Esempio:**

```
> x
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    1
> colnames(x)
NULL
> colnames(x)<-c("c1","c2","c3")
> x
     c1 c2 c3
[1,]  1  3  5
[2,]  2  4  1
> colnames(x)
[1] "c1" "c2" "c3"
```

dimnames()

- **Package:** `base`
- **Parametri:**

  `x` matrice di dimensione $n \times m$

- **Significato:** etichette di riga e di colonna
- **Esempio:**

```
> x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> dimnames(x)
NULL
> dimnames(x)<-list(c("r1","r2","r3"),c("c1","c2","c3"))
```

```
> x
   c1 c2 c3
r1  1  4  7
r2  2  5  8
r3  3  6  9
> dimnames(x)
[[1]]
[1] "r1" "r2" "r3"

[[2]]
[1] "c1" "c2" "c3"
```

[ ]

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** estrazione di elementi da una matrice
- **Esempio:**

```
> A
   c1 c2 c3
r1  1  4  7
r2  2  5  8
r3  3  6  8
> n<-3
> m<-3
> # elemento di posto (2,3)
> A[2,3]
[1] 8
> # prima riga
> A[1,]
c1 c2 c3
 1  4  7
> A["r1",]
c1 c2 c3
 1  4  7
> # terza colonna
> A[,3]
r1 r2 r3
 7  8  8
> A[,"c3"]
r1 r2 r3
 7  8  8
> # prima e seconda riga
> A[c(1,2),]
   c1 c2 c3
r1  1  4  7
r2  2  5  8
> A[c("r1","r2"),]
   c1 c2 c3
r1  1  4  7
r2  2  5  8
> # seconda e terza colonna
> A[,c(2,3)]
   c2 c3
r1  4  7
r2  5  8
r3  6  8
```

```
> A[,c("c2","c3")]
   c2 c3
r1  4  7
r2  5  8
r3  6  8
> # tutte le righe eccetto la prima
> A[-1,]
   c1 c2 c3
r2  2  5  8
r3  3  6  8
> # tutte le colonne eccetto la terza
> A[,-3]
   c1 c2
r1  1  4
r2  2  5
r3  3  6
> # tutte le righe che presentano un
> # valore maggiore di 4.1 nella colonna 2
> A[A[,"c2"]>4.1,]
   c1 c2 c3
r2  2  5  8
r3  3  6  8
> # tutti gli elementi
> # superiori a 3
> x[x>3]
[1] 4 5 6 7 8 9
```

col()

- **Package:** `base`
- **Parametri:**

  `data` matrice di dimensione $n \times m$

- **Significato:** colonna di appartenenza di ogni elemento
- **Esempio:**

```
>x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
>n<-3
>m<-3
>col(x)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    1    2    3
[3,]    1    2    3

>x
     [,1] [,2] [,3]
[1,]  1.1  4.5  8.8
[2,]  2.3  6.7  6.1
>n<-2
>m<-3
>col(x)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    1    2    3
```

row()

- **Package:** `base`
- **Parametri:**

  `data` matrice di dimensione $n \times m$

- **Significato:** riga di appartenenza di ogni elemento
- **Esempio:**

```
> x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> n<-3
> m<-3
> row(x)
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    2    2
[3,]    3    3    3

> x
     [,1] [,2] [,3]
[1,]  1.1  4.5  8.8
[2,]  2.3  6.7  6.1
> n<-2
> m<-3
> row(x)
     [,1] [,2] [,3]
[1,]    1    1    1
[2,]    2    2    2
```

head()

- **Package:** `utils`
- **Parametri:**

  `data` matrice di dimensione $k \times m$

  `n` numero di righe

- **Significato:** seleziona le prime n righe
- **Esempio:**

```
> x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> k<-3
> m<-3
> head(x,n=2)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8

> x
     [,1] [,2] [,3]
[1,]    1    2    3
```

```
[2,]    4    5    6
[3,]    7    8    9
> k<-3
> m<-3
> head(x,n=2)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

tail()

- **Package:** `utils`
- **Parametri:**

  `data` matrice di dimensione $k \times m$

  `n` numero di righe

- **Significato:** seleziona le ultime n righe
- **Esempio:**

```
> x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> k<-3
> m<-3
> tail(x,n=2)
     [,1] [,2] [,3]
[2,]    2    5    8
[3,]    3    6    9

> x
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> k<-3
> m<-3
> tail(x,n=2)
     [,1] [,2] [,3]
[2,]    4    5    6
[3,]    7    8    9
```

vech()

- **Package:** `MCMCpack`
- **Parametri:**

  `x` matrice di dimensione $m \times n$

- **Significato:** seleziona gli elementi della sezione triangolare inferiore di una matrice simmetrica
- **Esempio:**

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

```
[3,]    3    5    7    8
[4,]    4    6    8    9
> vech(x)
[1] 1 2 3 4 4 5 6 7 8 9

> x
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   12   14   15
[3,]   13   15   16
> vech(x)
[1] 11 12 13 14 15 16
```

xpnd()

- **Package:** `MCMCpack`
- **Parametri:**

  `x` vettore numerico di dimensione $n(n+1)/2$

  `nrow` numero n di righe

- **Significato:** crea una matrice simmetrica a partire da un vettore
- **Esempio:**

```
> xpnd(x=c(1,2,3,4,4,5,6,7,8,9),nrow=4)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    2    4    5    6
[3,]    3    5    7    8
[4,]    4    6    8    9

> xpnd(x=c(11,12,13,14,15,16),nrow=3)
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   12   14   15
[3,]   13   15   16
```

length()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** numero di elementi
- **Formula:**

$$n\,m$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> n<-3
> m<-3
> n*m
```

```
[1] 9
> length(A)
[1] 9

> A
     [,1] [,2]
[1,]  1.2  2.3
[2,]  4.5  3.1
> n<-2
> m<-2
> n*m
[1] 4
> length(A)
[1] 4
```

cbind()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

  `B` matrice di dimensione $n \times k$
- **Significato:** unisce due matrici accostandole per colonna
- **Esempio:**

```
> A
     [,1]
[1,]  9.9
[2,]  1.0
[3,] 12.0
> B
     [,1]
[1,]    1
[2,]    2
[3,]    3
> n<-3
> m<-1
> k<-1
> cbind(A,B)
     [,1] [,2]
[1,]  9.9    1
[2,]  1.0    2
[3,] 12.0    3

> A
     [,1]
[1,]    1
[2,]    2
> B
     [,1]
[1,]    3
[2,]    4
> n<-2
> m<-1
> k<-1
> cbind(A,B)
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

rbind()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

  `B` matrice di dimensione $k \times m$

- **Significato:** unisce due matrici accostandole per riga
- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  9.9    1   12
> B
     [,1] [,2] [,3]
[1,]    1    2    3
> n<-1
> m<-3
> k<-1
> rbind(A,B)
     [,1] [,2] [,3]
[1,]  9.9    1   12
[2,]  1.0    2    3
```

```
> A
     [,1]
[1,]    1
[2,]    2
> B
     [,1]
[1,]    3
[2,]    4
> n<-2
> m<-1
> k<-2
> rbind(A,B)
     [,1]
[1,]    1
[2,]    2
[3,]    3
[4,]    4
```

toeplitz()

- **Package:** `stats`
- **Parametri:**

  `data` vettore numerico di dimensione n

- **Significato:** matrice simmetrica di *Toeplitz* di dimensione $n \times n$
- **Esempio:**

```
> x
[1] 1 2 3
> n<-length(x)
> n
[1] 3
> toeplitz(x)
     [,1] [,2] [,3]
```

```
[1,]    1    2    3
[2,]    2    1    2
[3,]    3    2    1

> # matrice di Toeplitz di dimensione d x d
> # sulle autocorrelazioni di ordine d-1 di una serie storica
> x
[1] -2.05 -1.04  0.92 -0.67  0.82  0.09 -0.64  0.21  0.02  1.83
> d<-3
> rho<-as.vector(acf(x,lag=d-1,plot=F)$acf)
> rho
[1]  1.000000000 -0.007736872 -0.054134090
> toeplitz(rho)
             [,1]         [,2]         [,3]
[1,]  1.000000000 -0.007736872 -0.054134090
[2,] -0.007736872  1.000000000 -0.007736872
[3,] -0.054134090 -0.007736872  1.000000000
```

2.3 Operazioni sulle Matrici

det()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times n$

- **Significato:** determinante
- **Formula:**

$$\det(A)$$

- **Esempio:**

```
> A
     [,1] [,2]
[1,]  1.0 -0.2
[2,]  4.0  5.6
> n<-2
> det(A)
[1] 6.4

> A
     [,1] [,2] [,3]
[1,]  1.2  6.5  2.3
[2,]  2.3  7.6  4.5
[3,]  4.5  1.1  6.7
> n<-3
> det(A)
[1] 13.783
```

determinant()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times n$

  `logarithm = T / F` logaritmo naturale del modulo del determinante

- **Significato:** determinante
- **Output:**

  `modulus` modulo

  `sign` segno

- **Formula:**

  `logarithm = T`

  `modulus`

  $$\log\left(|\det(A)|\right)$$

  `sign`

  $$\operatorname{sign}\left(\det(A)\right)$$

  `logarithm = F`

  `modulus`

  $$|\det(A)|$$

  `sign`

  $$\operatorname{sign}\left(\det(A)\right)$$

- **Esempio:**

```
> A
     [,1] [,2]
[1,]  1.0 -0.2
[2,]  4.0  5.6
> n<-2
> abs(det(A))
[1] 6.4
> determinant(A,logarithm=F)$modulus
[1] 6.4
attr(,"logarithm")
[1] FALSE
> sign(det(A))
[1] 1
> determinant(A,logarithm=F)$sign
[1] 1

> A
     [,1] [,2] [,3]
[1,]  1.2  8.9  7.8
[2,]  4.5  4.5  7.5
[3,]  6.7  6.6  3.3
> n<-3
> abs(det(A))
[1] 269.97
> determinant(A,logarithm=F)$modulus
[1] 269.97
attr(,"logarithm")
[1] FALSE
> sign(det(A))
[1] 1
> determinant(A,logarithm=F)$sign
[1] 1
```

determinant.matrix()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times n$

  `logarithm = T / F` logaritmo naturale del modulo del determinante

- **Significato:** determinante
- **Output:**

  `modulus` modulo

  `sign` segno

- **Formula:**

  `logarithm = T`

  `modulus`

  $$\log\left(|\det(A)|\right)$$

  `sign`

  $$\operatorname{sign}\left(\det(A)\right)$$

  `logarithm = F`

  `modulus`

  $$|\det(A)|$$

  `sign`

  $$\operatorname{sign}\left(\det(A)\right)$$

- **Esempio:**

```
> A
     [,1] [,2]
[1,]  1.0 -0.2
[2,]  4.0  5.6
> n<-2
> abs(det(A))
[1] 6.4
> determinant.matrix(A,logarithm=F)$modulus
[1] 6.4
attr(,"logarithm")
[1] FALSE
> sign(det(A))
[1] 1
> determinant.matrix(A,logarithm=F)$sign
[1] 1

> A
     [,1] [,2] [,3]
[1,]  1.2  8.9  7.8
[2,]  4.5  4.5  7.5
[3,]  6.7  6.6  3.3
> n<-3
> abs(det(A))
[1] 269.97
> determinant.matrix(A,logarithm=F)$modulus
[1] 269.97
attr(,"logarithm")
[1] FALSE
> sign(det(A))
[1] 1
> determinant.matrix(A,logarithm=F)$sign
[1] 1
```

as.vector()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** trasforma la matrice in vettore di dimensione nm seguendo l'ordine delle colonne
- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> n<-3
> m<-3
> as.vector(A)
[1] 1 2 3 4 5 6 7 8 9

> A
     [,1] [,2]
[1,]  1.2  6.5
[2,]  2.3  7.6
> n<-2
> m<-2
> as.vector(A)
[1] 1.2 2.3 6.5 7.6
```

norm()

- **Package:** `Matrix`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

  `type = o / i / F / m` massima somma assoluta di colonna, massima somma assoluta di riga, norma di *Frobenius*, massimo valore assoluto

- **Significato:** norma
- **Formula:**

`type = o`

$$\max\left(\sum_{i=1}^{n} |\, a_{ij} \,|\right) \quad \forall j = 1, 2, \ldots, m$$

`type = i`

$$\max\left(\sum_{j=1}^{m} |\, a_{ij} \,|\right) \quad \forall i = 1, 2, \ldots, n$$

`type = F`

$$\left(\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}^2\right)^{1/2}$$

`type = m`

$$\max(|a_{ij}|) \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, m$$

- **Esempio:**

```
> n<-2
> m<-2
> x<-c(1.2,3.4,0.2,-1.2)
> A<-Matrix(x,nrow=n,ncol=m,byrow=F)
> A
2 x 2 Matrix of class 'dgeMatrix'
     [,1] [,2]
[1,]  1.2  0.2
[2,]  3.4 -1.2
> sqrt(1.2**2+0.2**2+3.4**2+(-1.2)**2)
[1] 3.805260
> norm(A,type="F")
[1] 3.805260
> max(abs(1.2),abs(0.2),abs(3.4),abs(-1.2))
[1] 3.4
> norm(A,type="m")
[1] 3.4
```

solve()

- **Package:** `base`
- **Parametri:**

  `A` matrice invertibile di dimensione $n \times n$

  `B` matrice di dimensione $n \times k$

- **Significato:** matrice inversa oppure soluzione di un sistema quadrato lineare
- **Formula:**

$$A^{-1} \qquad A^{-1}\,B$$

- **Esempio:**

```
> A
     [,1] [,2]
[1,]  1.0  4.0
[2,] -0.2  5.6
> n<-2
> invA<-solve(A)
> A%*%invA
             [,1] [,2]
[1,] 1.000000e+00    0
[2,] 1.109952e-17    1
> invA%*%A
            [,1]         [,2]
[1,] 1.00000e+00 2.220446e-16
[2,] 5.20417e-18 1.000000e+00

> A
     [,1] [,2]
[1,]  1.0  4.0
[2,] -0.2  5.6
> B
[1] 11 -2
> n<-2
> k<-1
```

```
> solve(A,B)
[1] 10.87500  0.03125
> solve(A)%*%B
         [,1]
[1,] 10.87500
[2,]  0.03125

> A
     [,1] [,2]
[1,]  1.0  4.0
[2,] -0.2  5.6
> B
     [,1] [,2]
[1,]   11 13.0
[2,]   -2  4.1
> n<-2
> k<-2
> solve(A,B)
         [,1]     [,2]
[1,] 10.87500 8.812500
[2,]  0.03125 1.046875
```

eigen()

- **Package:** `base`
- **Parametri:**

  `A` matrice simmetrica di dimensione $n \times n$

  `only.values = T / F` calcola i soli autovalori

- **Significato:** autovalori ed autovettori
- **Output:**

  `values` la diagonale della matrice D degli autovalori di dimensione $n \times n$

  `vectors` matrice ortogonale Γ degli autovettori di dimensione $n \times n$

- **Formula:**

$$A = \Gamma D \Gamma^T$$

$$\text{dove} \quad \Gamma^T \Gamma = I_n = \Gamma \Gamma^T \quad \text{e} \quad D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.2  3.0  5.6
[2,]  3.0  4.0  6.7
[3,]  5.6  6.7  9.8
> n<-3
> # A simmetrica
> D<-diag(eigen(A)$values)
> D
         [,1]       [,2]      [,3]
[1,] 16.77455  0.0000000  0.000000
[2,]  0.00000 -0.1731794  0.000000
[3,]  0.00000  0.0000000 -1.601373
> GAMMA<-eigen(A)$vectors
> GAMMA
           [,1]       [,2]       [,3]
```

```
[1,] -0.3767594  0.3675643  0.8502640
[2,] -0.4980954 -0.8542951  0.1485966
[3,] -0.7809951  0.3675274 -0.5049458
> GAMMA%*%D%*%t(GAMMA)
     [,1] [,2] [,3]
[1,]  1.2  3.0  5.6
[2,]  3.0  4.0  6.7
[3,]  5.6  6.7  9.8
> # A = GAMMA%*%D%*%t(GAMMA)

> A
     [,1] [,2]
[1,]  1.2  2.3
[2,]  2.3  2.2
> n<-2
> # A simmetrica
> D<-diag(eigen(A)$values)
> D
         [,1]       [,2]
[1,] 4.053720  0.0000000
[2,] 0.000000 -0.6537205
> GAMMA<-eigen(A)$vectors
> GAMMA
         [,1]      [,2]
[1,] 0.627523  0.778598
[2,] 0.778598 -0.627523
> GAMMA%*%D%*%t(GAMMA)
     [,1] [,2]
[1,]  1.2  2.3
[2,]  2.3  2.2
> # A = GAMMA%*%D%*%t(GAMMA)
```

crossprod()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

  `B` matrice di dimensione $n \times k$

- **Significato:** prodotto scalare
- **Formula:**

$$A^T A \qquad A^T B$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.2  3.0  5.6
[2,]  3.0  4.0  6.7
[3,]  5.6  6.7  9.8
> n<-3
> m<-3
> t(A)%*%A
      [,1]   [,2]   [,3]
[1,] 41.80  53.12  81.70
[2,] 53.12  69.89 109.26
[3,] 81.70 109.26 172.29
> crossprod(A)
      [,1]   [,2]   [,3]
```

```
[1,] 41.80  53.12  81.70
[2,] 53.12  69.89 109.26
[3,] 81.70 109.26 172.29

> A
     [,1] [,2] [,3]
[1,]  1.2  3.0  5.6
[2,]  3.0  4.0  6.7
[3,]  5.6  6.7  9.8
> B
     [,1] [,2]
[1,] 11.0  4.1
[2,] -2.0  5.0
[3,]  3.4  7.0
> n<-3
> m<-3
> k<-2
> t(A)%*%B
      [,1]   [,2]
[1,] 26.24  59.12
[2,] 47.78  79.20
[3,] 81.52 125.06
> crossprod(A,B)
      [,1]   [,2]
[1,] 26.24  59.12
[2,] 47.78  79.20
[3,] 81.52 125.06
```

tcrossprod()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

  `B` matrice di dimensione $k \times m$

- **Significato:** prodotto scalare
- **Formula:**

$$A\,A^T \qquad A\,B^T$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.2  3.0  5.6
[2,]  3.0  4.0  6.7
[3,]  5.6  6.7  9.8
> n<-3
> m<-3
> A%*%t(A)
      [,1]   [,2]   [,3]
[1,] 41.80  53.12  81.70
[2,] 53.12  69.89 109.26
[3,] 81.70 109.26 172.29
> tcrossprod(A)
      [,1]   [,2]   [,3]
[1,] 41.80  53.12  81.70
[2,] 53.12  69.89 109.26
[3,] 81.70 109.26 172.29
```

```
> A
     [,1] [,2] [,3]
[1,]  1.2  3.0  5.6
[2,]  3.0  4.0  6.7
[3,]  5.6  6.7  9.8
> B
     [,1] [,2] [,3]
[1,] 11.0   -2  3.4
[2,]  4.1    5  7.0
> n<-3
> m<-3
> k<-2
> A%*%t(B)
      [,1]   [,2]
[1,] 26.24  59.12
[2,] 47.78  79.20
[3,] 81.52 125.06
> tcrossprod(A,B)
      [,1]   [,2]
[1,] 26.24  59.12
[2,] 47.78  79.20
[3,] 81.52 125.06
```

*

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

  `B` matrice di dimensione $n \times m$

- **Significato:** prodotto di *Hadamard*
- **Formula:**

$$x_i \, y_j \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, m$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> B
     [,1] [,2] [,3]
[1,]  1.1  5.4  2.1
[2,]  2.3  4.6  3.2
[3,]  4.1  4.2  4.3
> n<-3
> m<-3
> A*B
     [,1] [,2] [,3]
[1,]  1.1 21.6 14.7
[2,]  4.6 23.0 25.6
[3,] 12.3 25.2 38.7

> A
     [,1] [,2]
[1,]    1    3
[2,]    2    5
> B
     [,1] [,2]
```

```
[1,]  1.1  4.5
[2,]  2.3  6.7
> n<-2
> m<-2
> A*B
     [,1] [,2]
[1,]  1.1 13.5
[2,]  4.6 33.5
```

%*%

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

  `B` matrice di dimensione $m \times k$

- **Significato:** prodotto scalare
- **Formula:**

$$A\,B$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> B
     [,1] [,2]
[1,] 11.0  4.1
[2,] -2.0  5.0
[3,]  3.4  7.0
> n<-3
> m<-3
> k<-2
> A%*%B
       [,1]   [,2]
[1,]  36.66  93.40
[2,] -10.00  34.18
[3,]  58.20 135.30
```

```
> A
     [,1] [,2]
[1,]    1    2
> B
     [,1]
[1,]    3
[2,]    4
> n<-1
> m<-2
> k<-1
> A%*%B
     [,1]
[1,]   11
```

kronecker()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

  `B` matrice di dimensione $h \times k$

- **Significato:** prodotto di *Kronecker*
- **Formula:**

$$A \otimes B = \begin{pmatrix} a_{1,1}\,B & \cdots & a_{1,m}\,B \\ \vdots & \vdots & \vdots \\ a_{n,1}\,B & \cdots & a_{n,m}\,B \end{pmatrix}$$

- **Esempio:**

```
> A
     [,1]
[1,]    1
[2,]    2
[3,]    3
> B
     [,1] [,2] [,3]
[1,]    7    8    9
> n<-3
> m<-1
> h<-1
> k<-3
> kronecker(A,B)
     [,1] [,2] [,3]
[1,]    7    8    9
[2,]   14   16   18
[3,]   21   24   27

> A
     [,1] [,2]
[1,]    1    2
> B
     [,1]
[1,]    3
[2,]    4
> n<-1
> m<-2
> h<-2
> k<-1
> kronecker(A,B)
     [,1] [,2]
[1,]    3    6
[2,]    4    8
```

diag()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times n$

  `x` vettore numerico di dimensione n

  `h` valore naturale

- **Significato:** estrae gli elementi diagonali o crea una matrice diagonale

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> n<-3
> diag(A)
[1] 1 5 9

> x<-1:3
> diag(x)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    2    0
[3,]    0    0    3

> h<-2
> diag(h)
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

t()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** trasposta
- **Formula:**

$$A^T$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,] 1.20  1.0 4.60
[2,] 3.40  2.0 7.80
[3,] 4.23  3.4 9.88
> n<-3
> m<-3
> t(A)
     [,1] [,2] [,3]
[1,]  1.2  3.4 4.23
[2,]  1.0  2.0 3.40
[3,]  4.6  7.8 9.88

> A
     [,1] [,2]
[1,]    1    2
> n<-1
> m<-2
> t(A)
     [,1]
[1,]    1
[2,]    2
```

aperm()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** trasposta
- **Formula:**

$$A^T$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,] 1.20  1.0 4.60
[2,] 3.40  2.0 7.80
[3,] 4.23  3.4 9.88
> n<-3
> m<-3
> aperm(A)
     [,1] [,2] [,3]
[1,]  1.2  3.4 4.23
[2,]  1.0  2.0 3.40
[3,]  4.6  7.8 9.88

> A
     [,1] [,2]
[1,]    1    2
> n<-1
> m<-2
> t(A)
     [,1]
[1,]    1
[2,]    2
```

dim()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** numero di righe e di colonne
- **Formula:**

$$n \quad m$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> dim(A)
[1] 3 3

> A
     [,1] [,2]
[1,]  1.2  6.5
[2,]  2.3  7.6
```

```
> n<-2
> m<-2
> dim(A)
[1] 2 2
```

nrow()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** numero di righe
- **Formula:**

$$n$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> nrow(A)
[1] 3
```

```
> A
     [,1] [,2]
[1,]  1.2  6.5
[2,]  2.3  7.6
> nrow(A)
[1] 2
```

NROW()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** numero di righe
- **Formula:**

$$n$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> NROW(A)
[1] 3
```

```
> A
     [,1] [,2]
[1,]  1.2  6.5
[2,]  2.3  7.6
> NROW(A)
[1] 2
```

ncol()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** numero di colonne
- **Formula:**

$$m$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> ncol(A)
[1] 3

> A
     [,1] [,2]
[1,]    1    2
> ncol(A)
[1] 2
```

NCOL()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** numero di colonne
- **Formula:**

$$m$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> NCOL(A)
[1] 3

> A
     [,1] [,2]
[1,]    1    2
> NCOL(A)
[1] 2
```

rowSums()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** somme di riga
- **Formula:**

$$\sum_{j=1}^{m} x_{ij} \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> n<-3
> m<-3
> rowSums(A)
[1] 14.9  6.4 22.8

> A
     [,1] [,2]
[1,]  1.2  4.5
[2,]  3.4  5.6
> n<-2
> m<-2
> rowSums(A)
[1] 5.7 9.0
```

rowMeans()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** medie di riga
- **Formula:**

$$\frac{1}{m} \sum_{j=1}^{m} x_{ij} \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> n<-3
> m<-3
> rowMeans(A)
[1] 4.966667 2.133333 7.600000

> A
     [,1] [,2]
[1,]  1.2  4.5
```

```
[2,]  3.4  5.6
> n<-2
> m<-2
> rowMeans(A)
[1] 2.85 4.50
```

colSums()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** somme di colonna
- **Formula:**

$$\sum_{i=1}^{n} x_{ij} \quad \forall j = 1, 2, \ldots, m$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> n<-3
> m<-3
> colSums(A)
[1]  3.8 17.4 22.9
```

```
> A
     [,1] [,2]
[1,]  1.2  4.5
[2,]  3.4  5.6
> n<-2
> m<-2
> colSums(A)
[1]  4.6 10.1
```

colMeans()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** medie di colonna
- **Formula:**

$$\frac{1}{n} \sum_{i=1}^{n} x_{ij} \quad \forall j = 1, 2, \ldots, m$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
```

```
> n<-3
> m<-3
> colMeans(A)
[1] 1.266667 5.800000 7.633333

> A
     [,1] [,2]
[1,]  1.2  4.5
[2,]  3.4  5.6
> n<-2
> m<-2
> colMeans(A)
[1] 2.30 5.05
```

rowsum()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

  `group` fattore f a k livelli di dimensione n

- **Significato:** applica la funzione somma ad ogni gruppo di elementi in ciascuna colonna di A definito dai livelli di f
- **Esempio:**

```
> A
     [,1] [,2]
[1,]  1.2  4.2
[2,]  2.3  2.1
[3,]  4.3  2.2
[4,]  4.2  4.0
> n<-4
> m<-2
> f
[1] 1 2 1 2
Levels: 1 2
> k<-nlevels(f)
> k
[1] 2
> rowsum(A,f)
  [,1] [,2]
1  5.5  6.4
2  6.5  6.1

> A
     [,1] [,2]
[1,]    1    7
[2,]    2    8
[3,]    3    9
[4,]    4    8
> n<-4
> m<-2
> k<-nlevels(f)
> k
[1] 3
> rowsum(A,f)
  [,1] [,2]
1    1    7
2    5   17
3    4    8
```

apply()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

  `MARGIN = 1 / 2` riga o colonna

  `FUN` funzione scelta

- **Significato:** applica `FUN` ad ogni riga o colonna della matrice A
- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> n<-3
> m<-3
> # medie di riga
> apply(A,MARGIN=1,FUN=mean)
[1] 4.966667 2.133333 7.600000

> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> n<-3
> m<-3
> # medie di colonna
> apply(A,MARGIN=2,FUN=mean)
[1] 1.266667 5.800000 7.633333
```

solveCrossprod()

- **Package:** `strucchange`
- **Parametri:**

  `A` matrice di dimensione $n \times k$ di rango $k = \min(n, k)$

  `method = qr / chol / solve` algoritmo risolutivo

- **Significato:** inversa del prodotto incrociato di X
- **Formula:**

$$(A^T A)^{-1}$$

- **Esempio:**

```
> A
     [,1] [,2]
[1,] 11.0  4.1
[2,] -2.0  5.0
[3,]  3.4  7.0
> n<-3
> k<-2
> A
     [,1] [,2]
```

```
[1,] 11.0  4.1
[2,] -2.0  5.0
[3,]  3.4  7.0
> solve(t(A)%*%A)
             [,1]         [,2]
[1,]  0.010167039 -0.006594413
[2,] -0.006594413  0.015289185
> solveCrossprod(A,method="qr")
             [,1]         [,2]
[1,]  0.010167039 -0.006594413
[2,] -0.006594413  0.015289185

> A
     [,1] [,2]
[1,]    1    7
[2,]    2    8
[3,]    3    9
[4,]    4    8
> n<-4
> k<-2
> solve(t(A)%*%A)
            [,1]        [,2]
[1,]  0.25393701 -0.08070866
[2,] -0.08070866  0.02952756
> solveCrossprod(A,method="qr")
            [,1]        [,2]
[1,]  0.25393701 -0.08070866
[2,] -0.08070866  0.02952756
```

model.matrix()

- **Package:** `base`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** matrice del modello di regressione lineare di dimensione $n \times k$
- **Formula:**

$$X = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{1,k-1} \\ 1 & x_{2,1} & \dots & x_{2,k-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,k-1} \end{pmatrix}$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> k<-4
> n<-length(y)
> X<-model.matrix(object=modello)
```

kappa()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** calcola il *ConditionNumber* come rapporto tra il maggiore ed il minore valore singolare non nullo della matrice diagonale D

- **Formula:**

$$\frac{\max\left(\text{diag}(D)\right)}{\min\left(\text{diag}(D)\right)}$$

$$\text{dove} \quad A = U\,D\,V^T \quad \text{e} \quad U^T\,U = I_k = V^T\,V = V\,V^T$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.2  3.0  5.6
[2,]  3.0  4.0  6.7
[3,]  5.6  6.7  9.8
> n<-3
> m<-3
> D<-diag(svd(A)$d)
> max(diag(D))/min(diag(D))
[1] 96.86229
> kappa(A,exact=T)
[1] 96.86229

> A
     [,1] [,2]
[1,]    1    7
[2,]    2    8
[3,]    3    9
[4,]    4    8
> n<-4
> m<-2
> D<-diag(svd(A)$d)
> max(diag(D))/min(diag(D))
[1] 8.923297
> kappa(A,exact=T)
[1] 8.923297
```

- **Osservazioni:** Calcola il *ConditionNumber* con la funzione `svd()`.

lower.tri()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times n$

- **Significato:** matrice triangolare inferiore di dimensione $n \times n$ a partire dalla matrice A
- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> n<-3
> A[t(lower.tri(A,diag=F))]<-0
> A
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    2    5    0
[3,]    3    6    9
```

```
> A
     [,1] [,2]
[1,]    1    7
[2,]    2    8
> n<-2
> A[t(lower.tri(A,diag=F))]<-0
> A
     [,1] [,2]
[1,]    1    0
[2,]    2    8
```

upper.tri()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times n$

- **Significato:** matrice triangolare superiore di dimensione $n \times n$ a partire dalla matrice A
- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> n<-3
> A[lower.tri(A,diag=F)]<-0
> A
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    0    5    8
[3,]    0    0    9
```

```
> A
     [,1] [,2]
[1,]    1    7
[2,]    2    8
> n<-2
> A[lower.tri(A,diag=F)]<-0
> A
     [,1] [,2]
[1,]    1    7
[2,]    0    8
```

backsolve()

- **Package:** `base`
- **Parametri:**

  `r` matrice A dei coefficienti di dimensione $n \times n$

  `data` matrice b dei termini noti di dimensione $1 \times n$

  `upper.tri = T / F` sistema triangolare superiore od inferiore

  `transpose = T / F` matrice dei coefficienti trasposta

- **Significato:** soluzione di un sistema triangolare di dimensione $n \times n$
- **Formula:**

`upper.tri = T` AND `transpose = T`

$$\left(\begin{array}{ccccc|c} a_{1,1} & 0 & \dots & \dots & 0 & b_1 \\ a_{1,2} & a_{2,2} & 0 & \dots & 0 & b_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ a_{1,n-1} & a_{2,n-1} & \dots & \ddots & 0 & \vdots \\ a_{1,n} & a_{2,n} & \dots & \dots & a_{n,n} & b_n \end{array}\right)$$

`upper.tri = T` AND `transpose = F`

$$\left(\begin{array}{ccccc|c} a_{1,1} & a_{1,2} & \dots & a_{1,n-1} & a_{1,n} & b_1 \\ 0 & a_{2,2} & \dots & a_{2,n-1} & a_{2,n} & b_2 \\ \vdots & 0 & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{n,n} & b_n \end{array}\right)$$

`upper.tri = F` AND `transpose = T`

$$\left(\begin{array}{ccccc|c} a_{1,1} & a_{2,1} & \dots & a_{n-1,1} & a_{n,1} & b_1 \\ 0 & a_{2,2} & \dots & a_{n-1,2} & a_{n,2} & b_2 \\ \vdots & 0 & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{n,n} & b_n \end{array}\right)$$

`upper.tri = F` AND `transpose = F`

$$\left(\begin{array}{ccccc|c} a_{1,1} & 0 & \dots & \dots & 0 & b_1 \\ a_{2,1} & a_{2,2} & 0 & \dots & 0 & b_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \dots & \ddots & 0 & \vdots \\ a_{n,1} & a_{n,2} & \dots & \dots & a_{n,n} & b_n \end{array}\right)$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> b
[1] 8 4 2
> backsolve(r=A,x=b,upper.tri=T,transpose=T)
[1]  8.000000 -5.000000 -6.016667
```

forwardsolve()

- **Package:** `base`
- **Parametri:**

  `r` matrice A dei coefficienti di dimensione $n \times n$

  `data` matrice b dei termini noti di dimensione $1 \times n$

  `upper.tri = T / F` sistema triangolare superiore od inferiore

  `transpose = T / F` matrice dei coefficienti trasposta

- **Significato:** soluzione di un sistema triangolare di dimensione $n \times n$

- **Formula:**

`upper.tri = T AND transpose = T`

$$\left(\begin{array}{ccccc|c} a_{1,1} & 0 & \dots & \dots & 0 & b_1 \\ a_{1,2} & a_{2,2} & 0 & \dots & 0 & b_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ a_{1,n-1} & a_{2,n-1} & \dots & \ddots & 0 & \vdots \\ a_{1,n} & a_{2,n} & \dots & \dots & a_{n,n} & b_n \end{array}\right)$$

`upper.tri = T AND transpose = F`

$$\left(\begin{array}{ccccc|c} a_{1,1} & a_{1,2} & \dots & a_{1,n-1} & a_{1,n} & b_1 \\ 0 & a_{2,2} & \dots & a_{2,n-1} & a_{2,n} & b_2 \\ \vdots & 0 & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{n,n} & b_n \end{array}\right)$$

`upper.tri = F AND transpose = T`

$$\left(\begin{array}{ccccc|c} a_{1,1} & a_{2,1} & \dots & a_{n-1,1} & a_{n,1} & b_1 \\ 0 & a_{2,2} & \dots & a_{n-1,2} & a_{n,2} & b_2 \\ \vdots & 0 & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{n,n} & b_n \end{array}\right)$$

`upper.tri = F AND transpose = F`

$$\left(\begin{array}{ccccc|c} a_{1,1} & 0 & \dots & \dots & 0 & b_1 \\ a_{2,1} & a_{2,2} & 0 & \dots & 0 & b_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \dots & \ddots & 0 & \vdots \\ a_{n,1} & a_{n,2} & \dots & \dots & a_{n,n} & b_n \end{array}\right)$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> b
[1] 8 4 2
> forwardsolve(r=A,x=b,upper.tri=T,transpose=T)
[1]  8.000000 -5.000000 -6.016667
```

2.4 Fattorizzazioni di Matrici

svd()

- **Package:** `base`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** fattorizzazione ai valori singolari

- **Output:**

  `d` diagonale della matrice D dei valori singolari di dimensione $m \times m$

  `u` matrice U di dimensione $n \times m$

  `v` matrice ortogonale V di dimensione $m \times m$

- **Formula:**

$$A = U\,D\,V^T$$

$$\text{dove} \quad U^T U = I_m = V^T V = V\,V^T$$

- **Esempio:**

```
> A
     [,1] [,2]
[1,] 11.0  4.1
[2,] -2.0  5.0
[3,]  3.4  7.0
> n<-3
> m<-2
> D<-diag(svd(A)$d)
> D
         [,1]     [,2]
[1,] 13.29929 0.000000
[2,]  0.00000 7.106262
> U<-svd(A)$u
> U
           [,1]       [,2]
[1,] -0.8566792  0.3981302
[2,] -0.0882360 -0.7395948
[3,] -0.5082471 -0.5426710
> t(U)%*%U
              [,1]          [,2]
[1,]  1.000000e+00 -3.762182e-17
[2,] -3.762182e-17  1.000000e+00
> V<-svd(A)$v
> V
           [,1]       [,2]
[1,] -0.8252352  0.5647893
[2,] -0.5647893 -0.8252352
> t(V)%*%V
              [,1]          [,2]
[1,]  1.000000e+00 -2.222614e-18
[2,] -2.222614e-18  1.000000e+00
> V%*%t(V)
             [,1]         [,2]
[1,] 1.000000e+00 2.222614e-18
[2,] 2.222614e-18 1.000000e+00
> U%*%D%*%t(V)
     [,1] [,2]
[1,] 11.0  4.1
[2,] -2.0  5.0
[3,]  3.4  7.0
> # A = U%*%D%*%t(V)

> A
     [,1] [,2]
[1,]    1 3.45
[2,]    2 7.80
> n<-2
> m<-2
> D<-diag(svd(A)$d)
```

```
> D
        [,1]      [,2]
[1,] 8.81658 0.0000000
[2,] 0.00000 0.1020804
> U<-svd(A)$u
> U
           [,1]       [,2]
[1,] -0.4072775 -0.9133044
[2,] -0.9133044  0.4072775
> t(U)%*%U
              [,1]          [,2]
[1,]  1.000000e+00 -2.201201e-16
[2,] -2.201201e-16  1.000000e+00
> V<-svd(A)$v
> V
           [,1]       [,2]
[1,] -0.2533734 -0.9673686
[2,] -0.9673686  0.2533734
> t(V)%*%V
             [,1]         [,2]
[1,] 1.000000e+00 3.401684e-18
[2,] 3.401684e-18 1.000000e+00
> V%*%t(V)
             [,1]         [,2]
[1,] 1.000000e+00 3.401684e-18
[2,] 3.401684e-18 1.000000e+00
> U%*%D%*%t(V)
     [,1] [,2]
[1,]    1 3.45
[2,]    2 7.80
> # A = U%*%D%*%t(V)
```

qr.Q()

- **Package:** `base`
- **Parametri:**

  `A` matrice di rango pieno di dimensione $n \times m$

- **Significato:** matrice Q di dimensione $n \times m$
- **Formula:**

$$A = Q\,R$$

$$\text{dove} \quad Q^T\,Q = I_m$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> n<-3
> m<-3
> Q<-qr.Q(qr(A))
> Q
             [,1]         [,2]       [,3]
[1,] -0.31559720 -0.220214186 -0.9229865
[2,]  0.06311944 -0.975415572  0.2111407
```

```
[3,] -0.94679160  0.008377024  0.3217382
> t(Q)%*%Q
              [,1]          [,2]          [,3]
[1,]  1.000000e+00 -1.690678e-17 -4.214836e-17
[2,] -1.690678e-17  1.000000e+00  3.281046e-17
[3,] -4.214836e-17  3.281046e-17  1.000000e+00

> A
     [,1] [,2]
[1,]    1 3.45
[2,]    2 7.80
> n<-2
> m<-2
> Q<-qr.Q(qr(A))
> Q
           [,1]       [,2]
[1,] -0.4472136 -0.8944272
[2,] -0.8944272  0.4472136
> t(Q)%*%Q
              [,1]          [,2]
[1,]  1.000000e+00 -1.260385e-17
[2,] -1.260385e-17  1.000000e+00
```

qr.R()

- **Package:** `base`
- **Parametri:**

  `A` matrice di rango pieno di dimensione $n \times m$

- **Significato:** matrice R triangolare superiore di dimensione $m \times m$
- **Formula:**

$$A = Q\,R$$

- **Esempio:**

```
> A
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> n<-3
> m<-3
> R<-qr.R(qr(A))
> R
          [,1]      [,2]       [,3]
[1,] -3.168596 -8.293894 -14.422792
[2,]  0.000000 -6.277843  -3.055012
[3,]  0.000000  0.000000  -5.065567
> Q<-qr.Q(qr(A))
> Q
             [,1]         [,2]       [,3]
[1,] -0.31559720 -0.220214186 -0.9229865
[2,]  0.06311944 -0.975415572  0.2111407
[3,] -0.94679160  0.008377024  0.3217382
> Q%*%R
     [,1] [,2] [,3]
[1,]  1.0  4.0  9.9
[2,] -0.2  5.6  1.0
[3,]  3.0  7.8 12.0
> # A = Q%*%R
```

```
> A
     [,1] [,2]
[1,]    1 3.45
[2,]    2 7.80
> n<-2
> m<-2
> R<-qr.R(qr(A))
> R
          [,1]       [,2]
[1,] -2.236068 -8.5194190
[2,]  0.000000  0.4024922
> Q<-qr.Q(qr(A))
> Q
           [,1]       [,2]
[1,] -0.4472136 -0.8944272
[2,] -0.8944272  0.4472136
> Q%*%R
     [,1] [,2]
[1,]    1 3.45
[2,]    2 7.80
> # A = Q%*%R
```

chol()

- **Package:** `base`
- **Parametri:**

  `A` matrice simmetrica definita positiva di dimensione $n \times n$

- **Significato:** matrice P triangolare superiore di dimensione $n \times n$
- **Formula:**

$$A = P^T P$$

- **Esempio:**

```
> A
     [,1] [,2]
[1,]    5    1
[2,]    1    3
> n<-2
> # A simmetrica definita positiva
> P<-chol(A)
> P
         [,1]      [,2]
[1,] 2.236068 0.4472136
[2,] 0.000000 1.6733201
> t(P)%*%P
     [,1] [,2]
[1,]    5    1
[2,]    1    3
> # A = t(P)%*%P

> A
     [,1] [,2]
[1,]  1.2  3.4
[2,]  3.4 11.2
> n<-2
> # A simmetrica definita positiva
> P<-chol(A)
> P
```

```
         [,1]     [,2]
[1,] 1.095445 3.103761
[2,] 0.000000 1.251666
> t(P)%*%P
     [,1] [,2]
[1,]  1.2  3.4
[2,]  3.4 11.2
> # A = t(P)%*%P
```

chol2inv()

- **Package:** `base`
- **Parametri:**

  `P` matrice P triangolare superiore di dimensione $n \times n$

- **Significato:** funzione inversa di `chol()`
- **Formula:**

$$(P^T P)^{-1}$$

- **Esempio:**

```
> A
     [,1] [,2]
[1,]    5    1
[2,]    1    3
> n<-2
> # A simmetrica definita positiva
> P<-chol(A)
> P
         [,1]      [,2]
[1,] 2.236068 0.4472136
[2,] 0.000000 1.6733201
> t(P)%*%P
     [,1] [,2]
[1,]    5    1
[2,]    1    3
> # A = t(P)%*%P
> chol2inv(P)
            [,1]        [,2]
[1,]  0.21428571 -0.07142857
[2,] -0.07142857  0.35714286
> solve(A)
            [,1]        [,2]
[1,]  0.21428571 -0.07142857
[2,] -0.07142857  0.35714286
> # solve(A) = chol2inv(P)

> A
     [,1] [,2]
[1,]  1.2  3.4
[2,]  3.4 11.2
> n<-2
> # A simmetrica definita positiva
> P<-chol(A)
> P
         [,1]     [,2]
[1,] 1.095445 3.103761
[2,] 0.000000 1.251666
> t(P)%*%P
     [,1] [,2]
```

```
[1,]  1.2  3.4
[2,]  3.4 11.2
> # A = t(P)%*%P
> chol2inv(P)
          [,1]       [,2]
[1,]  5.957447 -1.8085106
[2,] -1.808511  0.6382979
> solve(A)
          [,1]       [,2]
[1,]  5.957447 -1.8085106
[2,] -1.808511  0.6382979
> # solve(A) = chol2inv(P)
```

ginv()

- **Package:** `MASS`
- **Parametri:**

  `A` matrice di dimensione $n \times m$

- **Significato:** inversa generalizzata A_g di dimensione $m \times n$
- **Formula:**

$$A = A\,A_g\,A$$

- **Esempio:**

```
> A
     [,1] [,2]
[1,]  1.0  4.0
[2,] -0.2  5.6
[3,]  3.0  7.8
> n<-3
> m<-2
> Ag<-ginv(A)
> Ag
            [,1]       [,2]         [,3]
[1,] 0.007783879 -0.4266172  0.302297558
[2,] 0.035078001  0.1553743 -0.001334379
> A%*%Ag%*%A
     [,1] [,2]
[1,]  1.0  4.0
[2,] -0.2  5.6
[3,]  3.0  7.8
> # A = A%*%Ag%*%A

> A
     [,1] [,2]
[1,]  1.2  3.4
[2,]  3.4 11.2
> n<-2
> m<-2
> Ag<-ginv(A)
> Ag
          [,1]       [,2]
[1,]  5.957447 -1.8085106
[2,] -1.808511  0.6382979
> A%*%Ag%*%A
     [,1] [,2]
[1,]  1.2  3.4
[2,]  3.4 11.2
> # A = A%*%Ag%*%A
```

2.5 Creazione di Array

array()

- **Package:** base
- **Parametri:**

  `data` vettore numerico

  `dim` dimensione

  `dimnames` etichette di dimensione

- **Significato:** creazione
- **Esempio:**

```
> etichette<-list(c("A","B"),c("a","b"),c("X","Y"))
> prova<-array(data=1:8,dim=c(2,2,2),dimnames=etichette)
> prova
, , X

  a b
A 1 3
B 2 4

, , Y

  a b
A 5 7
B 6 8

> etichette<-list(c("A","B"),c("a","b"))
> x<-array(data=1:8,dim=c(2,2),dimnames=etichette)
> x
  a b
A 1 3
B 2 4

> x<-seq(1:12)
> dim(x)<-c(3,2,2)
> x
, , 1

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

, , 2

     [,1] [,2]
[1,]    7   10
[2,]    8   11
[3,]    9   12
```

dim()

- **Package:** base
- **Parametri:**

  `x` array

- **Significato:** dimensione
- **Esempio:**

```
> n<-3
> m<-3
> x<-1:9
> dim(x)<-c(n,m)
> x
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9

> x<-seq(1:12)
> dim(x)<-c(3,2,2)
> x
, , 1

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

, , 2

     [,1] [,2]
[1,]    7   10
[2,]    8   11
[3,]    9   12
```

[ ]

- **Package:** `base`
- **Parametri:**

  `x` array

- **Significato:** estrazione di elementi
- **Esempio:**

```
> x<-seq(1:12)
> dim(x)<-c(2,3,2)
> x
, , 1

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

, , 2

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

>  # prime due colonne della seconda sottomatrice
>  x[1,1:2,2]
[1] 7 9

> x[1,2:3,]
```

```
     [,1] [,2]
[1,]    3    9
[2,]    5   11

> x[1,2:3,,drop=F]
, , 1

     [,1] [,2]
[1,]    3    5

, , 2

     [,1] [,2]
[1,]    9   11
```

dimnames()

- **Package:** `base`
- **Parametri:**

  `x` array

- **Significato:** etichette di dimensione
- **Esempio:**

```
> x
, , 1

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

, , 2

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

> dimnames(x)<-list(letters[1:2],LETTERS[1:3],c("primo","secondo"))
> x
, , primo

  A B C
a 1 3 5
b 2 4 6

, , secondo

  A  B  C
a 7  9 11
b 8 10 12
```

Parte II

Statistica Descrittiva

Capitolo 3

Funzioni ed Indici statistici

3.1 Funzioni di base

length()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n
- **Significato:** dimensione campionaria
- **Formula:**

$$n$$

- **Esempio:**

```
> x
[1] 1.2 2.3 4.5 6.5
> length(x)
[1] 4

> x
[1] 1.2 3.4 4.5 6.4 4.0 3.0 4.0
> length(x)
[1] 7
```

min()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n
- **Significato:** minimo
- **Formula:**

$$x_{(1)}$$

- **Esempio:**

```
> x
[1] 1.2 2.3 4.5 6.5
> min(x)
[1] 1.2

> x
[1] 1.1 3.4 4.5 6.4 4.0 3.0 4.0
> min(x)
[1] 1.1
```

max()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** massimo
- **Formula:**

$$x_{(n)}$$

- **Esempio:**

```
> x
[1] 1.2 2.3 4.5 6.5
> max(x)
[1] 6.5

> x
[1] 1.1 3.4 4.5 6.4 4.0 3.0 4.0
> max(x)
[1] 6.4
```

3.2 Indici di posizione

mean()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `trim` il valore di α con $0 \leq \alpha \leq 0.5$ che rappresenta la percentuale di osservazioni più basse e più alte che deve essere esclusa dal calcolo della media aritmetica

- **Significato:** media α-*trimmed*
- **Formula:**

$$\bar{x}_\alpha = \begin{cases} \bar{x} & \text{se } \alpha = 0 \\ \frac{1}{n-2\lfloor n\,\alpha \rfloor} \sum_{i=\lfloor n\,\alpha \rfloor+1}^{n-\lfloor n\,\alpha \rfloor} x_{(i)} & \text{se } 0 < \alpha < 0.5 \\ Q_{0.5}(x) & \text{se } \alpha = 0.5 \end{cases}$$

- **Esempio:**

```
> x
[1]  1.00  1.20  3.40  0.80 10.20  9.30  7.34
> n<-length(x)
> n
[1] 7
> sum(x)/n
[1] 4.748571
> alpha<-0
> mean(x,trim=alpha)
[1] 4.748571

> x
[1]  1.00  1.20  3.40  0.80 10.20  9.30 7.34
> x<-sort(x,decreasing=F)
> # x ordinato in maniera crescente
```

```
> n<-length(x)
> n
[1] 7
> alpha<-0.26
> sum(x[(floor(n*alpha)+1):(n-floor(n*alpha))])/(n-2*floor(n*alpha))
[1] 4.448
> mean(x,trim=alpha)
[1] 4.448

> x
[1]  1.00  1.20  3.40  0.80 10.20  9.30  7.34
> median(x)
[1] 3.4
> alpha<-0.5
> mean(x,trim=alpha)
[1] 3.4
```

weighted.mean()

- **Parametri:**
- **Package:** `stats`

  `x` vettore numerico di dimensione n

  `w` vettore numerico di pesi di dimensione n

- **Significato:** media pesata
- **Formula:**

$$\bar{x}_W = \frac{\sum_{i=1}^{n} x_i\, w_i}{\sum_{j=1}^{n} w_j}$$

- **Esempio:**

```
> x
[1] 3.7 3.3 3.5 2.8
> w
[1] 5 5 4 1
> sum(w)
[1] 15
> sum(x*w)/sum(w)
[1] 3.453333
> weighted.mean(x,w)
[1] 3.453333

> x
[1] 3.7 3.3 3.5 2.8
> w
[1] 0.16 0.34 0.28 0.22
> sum(w)
[1] 1
> sum(x*w)
[1] 3.325
> weighted.mean(x,w)
[1] 3.325
```

mean.a()

- **Package:** `labstatR`
- **Parametri:**

`x` vettore numerico di elementi non nulli di dimensione n

- **Significato:** media armonica
- **Formula:**

$$\bar{x}_A = \left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{x_i} \right)^{-1}$$

- **Esempio:**

```
> x
[1] 1.2 2.3 4.5 6.5
> # x a valori non nulli
> 1/mean(1/x)
[1] 2.432817
> mean.a(x)
[1] 2.432817

> x
[1] 1.2 3.4 4.5 6.4 4.0 3.0 4.0
> # x a valori non nulli
> 1/mean(1/x)
[1] 2.992404
> mean.a(x)
[1] 2.992404
```

mean.g()

- **Package:** `labstatR`
- **Parametri:**

`x` vettore numerico di elementi positivi di dimensione n

- **Significato:** media geometrica
- **Formula:**

$$\bar{x}_G = \left(\prod_{i=1}^{n} x_i \right)^{1/n}$$

- **Esempio:**

```
> x
[1] 1.2 2.3 4.5 6.5
> n<-length(x)
> n
[1] 4
> # x a valori positivi
> prod(x)**(1/n)
[1] 2.997497
> mean.g(x)
[1] 2.997497

> x
[1] 1.2 3.4 4.5 6.4 4.0 3.0 4.0
> n<-length(x)
> n
[1] 7
> # x a valori positivi
> prod(x)**(1/n)
[1] 3.434782
> mean.g(x)
[1] 3.434782
```

3.3 Indici di variabilità

range()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** campo di variazione
- **Formula:**

$$x_{(1)} \quad x_{(n)}$$

- **Esempio:**

```
> x
[1] 1.0 1.2 3.4 0.8
> min(x)
[1] 0.8
> max(x)
[1] 3.4
> range(x)
[1] 0.8 3.4

> x
[1] 1.2 3.4 4.5 6.4 4.0 3.0 4.0
> min(x)
[1] 1.2
> max(x)
[1] 6.4
> range(x)
[1] 1.2 6.4
```

quantile()

- **Package:** `stats`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `probs` valore p di probabilità

- **Significato:** quantile al $(100\,p)\%$
- **Formula:**

$$Q_p(x) = \begin{cases} x_{(\alpha)} & \text{se } \alpha \text{ è intero} \\ x_{(\lfloor\alpha\rfloor)} + (\alpha - \lfloor\alpha\rfloor)\,\left(x_{(\lfloor\alpha\rfloor+1)} - x_{(\lfloor\alpha\rfloor)}\right) & \text{se } \alpha \text{ non è intero} \end{cases}$$

$$\text{dove} \quad \alpha = 1 + (n-1)\,p$$

- **Esempio:**

```
> x
[1] 1.20 2.30 0.11 4.50
> x<-sort(x,decreasing=F)
> # x ordinato in maniera crescente
> n<-length(x)
> n
[1] 4
> p<-1/3
```

```
> alpha<-1+(n-1)*p
> alpha
[1] 2
> # alpha intero
> x[alpha]
[1] 1.2
> quantile(x,probs=1/3)
33.33333%
      1.2

> x
[1] 1.20 2.30 0.11 4.50
> x<-sort(x,decreasing=F)
> # x ordinato in maniera crescente
> n<-length(x)
> n
[1] 4
> p<-0.34
> alpha<-1+(n-1)*p
> alpha
[1] 2.02
> # alpha non intero
> x[floor(alpha)]+(alpha-floor(alpha))*(x[floor(alpha)+1]-x[floor(alpha)])
[1] 1.222
> quantile(x,probs=p)
  34%
1.222
```

median()

- **Package:** `stats`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** mediana
- **Formula:**

$$Q_{0.5}(x) = \begin{cases} x_{(\frac{n+1}{2})} & \text{se } n \text{ è dispari} \\ 0.5 \left(x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)} \right) & \text{se } n \text{ è pari} \end{cases}$$

- **Esempio:**

```
> x
[1]  1.20  0.34  5.60  7.40  2.10  3.20  9.87 10.10
> x<-sort(x,decreasing=F)
> # x ordinato in maniera crescente
> x
[1]  0.34  1.20  2.10  3.20  5.60  7.40  9.87 10.10
> n<-length(x)
> n
[1] 8
> # n pari
> 0.5*(x[n/2]+x[n/2+1])
[1] 4.4
> median(x)
[1] 4.4

> x
[1] 1.20 0.34 5.60 7.40 2.10 3.20 9.87
```

```
> x<-sort(x,decreasing=F)
> # x ordinato in maniera crescente
> n<-length(x)
> n
[1] 7
> # n dispari
> x[(n+1)/2]
[1] 3.2
> median(x)
[1] 3.2
```

IQR()

- **Package:** `stats`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** range interquartile
- **Formula:**

$$IQR(x) = Q_{0.75}(x) - Q_{0.25}(x)$$

- **Esempio:**

```
> x
[1]  1.00  1.20  3.40  0.80 10.20  9.30  7.34
> diff(quantile(x,probs=c(0.25,0.75)))
 75%
7.22
> IQR(x)
[1] 7.22

> x
[1] 1.2 3.4 4.5 6.4 4.0 3.0 4.0
> diff(quantile(x,probs=c(0.25,0.75)))
 75%
1.05
> IQR(x)
[1] 1.05
```

- **Osservazioni:** Calcola i quartili con la funzione `quantile()`.

mad()

- **Package:** `stats`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `center` parametro rispetto al quale si effettuano gli scarti

  `constant` il valore della costante *const*

- **Significato:** deviazione assoluta dalla mediana
- **Formula:**

$$const \cdot Q_{0.5}\left(\,|\,x - \text{center}(x)\,|\,\right)$$

- **Esempio:**

```
> x
[1]  3  5 11 14 15 20 22
> const<-1.23
> const*median(abs(x-median(x)))
[1] 7.38
> mad(x,center=median(x),constant=const)
[1] 7.38

> x
[1]  3  5 11 14 15 20 22
> const<-1.23
> const*median(abs(x-mean(x)))
[1] 8.785714
> mad(x,center=mean(x),constant=const)
[1] 8.785714
```

cv()

- **Package:** `labstatR`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** coefficiente di variazione nella popolazione
- **Formula:**

$$cv_x = \frac{\sigma_x}{|\bar{x}|}$$

- **Esempio:**

```
> x
[1] 1.0 1.2 3.4 0.8
> sigmax<-sqrt(sigma2(x))
> sigmax/abs(mean(x))
[1] 0.6555055
> cv(x)
[1] 0.6555055

> x
[1] 1.2 3.4 4.5 6.4 4.0 3.0 4.0
> sigmax<-sqrt(sigma2(x))
> sigmax/abs(mean(x))
[1] 0.3852385
> cv(x)
[1] 0.3852385
```

cv2()

- **Package:** `sigma2tools`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** coefficiente di variazione campionario
- **Formula:**

$$cv_x = \frac{s_x}{|\bar{x}|}$$

- **Esempio:**

```
> x
[1] 1.0 1.2 3.4 0.8
> sd(x)/abs(mean(x))
[1] 0.7569126
> cv2(x)
[1] 0.7569126

> x
[1] 1.2 3.4 4.5 6.4 4.0 3.0 4.0
> sd(x)/abs(mean(x))
[1] 0.4161051
> cv2(x)
[1] 0.4161051
```

stderror()

- **Package:** `sigma2tools`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** errore standard campionario
- **Formula:**

$$se_x = \frac{s_x}{\sqrt{n}}$$

- **Esempio:**

```
> x
[1] 1.0 1.2 3.4 0.8
> n<-length(x)
> sd(x)/sqrt(n)
[1] 0.6055301
> stderror(x)
[1] 0.6055301

> x
[1] 1.2 3.4 4.5 6.4 4.0 3.0 4.0
> n<-length(x)
> sd(x)/sqrt(n)
[1] 0.5953905
> stderror(x)
[1] 0.5953905
```

popstderror()

- **Package:** `sigma2tools`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** errore standard nella popolazione
- **Formula:**

$$se_x = \frac{\sigma_x}{\sqrt{n}}$$

- **Esempio:**

```
> x
[1] 1.0 1.2 3.4 0.8
> n<-length(x)
> sigmax<-sqrt((n-1)/n*var(x))
> sigmax/sqrt(n)
[1] 0.5244044
> popstderror(x)
[1] 0.5244044

> x
[1] 1.2 3.4 4.5 6.4 4.0 3.0 4.0
> n<-length(x)
> sigmax<-sqrt((n-1)/n*var(x))
> sigma(x)/sqrt(n)
[1] 0.5512245
> popstderror(x)
[1] 0.5512245
```

ssdev()

- **Package:** `sigma2tools`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** devianza
- **Formula:**

$$ss_x = \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - n\,\bar{x}^2$$

- **Esempio:**

```
> x
[1] 1.0 1.2 3.4 0.8
> sum((x-mean(x))**2)
[1] 4.4
> ssdev(x)
[1] 4.4

> x
[1] 1.2 2.3 4.5 6.5
> sum((x-mean(x))**2)
[1] 16.6675
> ssdev(x)
[1] 16.6675
```

sigma()

- **Package:** `sigma2tools`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** scarto quadratico medio
- **Formula:**

$$\sigma_x = \left(\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right)^{1/2}$$

- **Esempio:**

```
> x
[1] 1.0 2.3 4.5 6.7 8.9
> sqrt(mean((x-mean(x))**2))
[1] 2.868031
> sigma(x)
[1] 2.868031

> x
[1] 1.2 2.3 4.5 6.5
> sqrt(mean((x-mean(x))**2))
[1] 2.041292
> sigma(x)
[1] 2.041292
```

sigma2()

- **Package:** `labstatR`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** varianza nella popolazione
- **Formula:**

$$\sigma_x^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

- **Esempio:**

```
> x
[1] 1.0 2.3 4.5 6.7 8.9
> mean((x-mean(x))**2)
[1] 8.2256
> sigma2(x)
[1] 8.2256

> x
[1] 1.2 2.3 4.5 6.5
> mean((x-mean(x))**2)
[1] 4.166875
> sigma2(x)
[1] 4.166875
```

sigma2m()

- **Package:** `sigma2tools`
- **Parametri:**

  `x` matrice di dimensione $n \times k$ le cui colonne corrispondono ai vettori numerici x_1, x_2, ..., x_k

- **Significato:** matrice di covarianza non corretta
- **Esempio:**

```
> x
     [,1] [,2]
[1,]  154  108
[2,]  109  115
```

```
[3,]  137  126
[4,]  115   92
[5,]  140  146
> sigma2m(x)
      [,1]   [,2]
[1,] 277.2 110.40
[2,] 110.4 326.24

> x
     [,1] [,2] [,3]
[1,]  1.2  6.4  4.0
[2,]  3.4  4.0  1.2
[3,]  4.5  3.0  4.5
> sigma2m(x)
            [,1]       [,2]        [,3]
[1,]  1.88222222 -1.9555556 -0.09777778
[2,] -1.95555556  2.0355556  0.19111111
[3,] -0.09777778  0.1911111  2.10888889
[1] 4.166875
```

var()

- **Package:** `stats`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** varianza campionaria
- **Formula:**

$$s_x^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

- **Esempio:**

```
> x
[1] 1.0 2.3 4.5 6.7 8.9
> n<-length(x)
> n
[1] 5
> sum((x-mean(x))**2)/(n-1)
[1] 10.282
> var(x)
[1] 10.282

> x
[1] 1.0 2.3 4.5 6.7 8.9
> y
[1] 1 3 4 6 8
> n<-length(x)
> n
[1] 5
> sum((x-mean(x))**2)/(n-1)
[1] 10.282
> sum((y-mean(y))**2)/(n-1)
[1] 7.3
> sum((x-mean(x))*(y-mean(y)))/(n-1)
[1] 8.585
> z<-cbind(x,y)
> var(z)
       x     y
x 10.282 8.585
y  8.585 7.300
```

Var()

- **Package:** `car`
- **Parametri:**

  `x` matrice di dimensione $n \times k$ le cui colonne corrispondono ai vettori numerici $x_1, x_2, \ldots, x_k$

  `diag = T / F` varianze campionarie o matrice di covarianza

- **Significato:** matrice di covarianza
- **Formula:**

`diag = T`

$$s^2_{x_i} = \frac{1}{n-1}(x_i - \bar{x}_i)^T (x_i - \bar{x}_i) \quad \forall i, = 1, 2, \ldots, k$$

`diag = F`

$$s_{x_i x_j} = \frac{1}{n-1}(x_i - \bar{x}_i)^T (x_j - \bar{x}_j) \quad \forall i, j = 1, 2, \ldots, k$$

- **Esempio:**

```
> k<-2
> x1
[1]  0.5 -0.1  0.2 -1.9  1.9  0.7 -1.5  0.0 -2.5  1.6  0.2 -0.3
> x2
[1]  1.0  4.0 10.0  2.1  3.5  5.6  8.4 12.0  6.5  2.0  1.2  3.4
> n<-length(x1)
> n
[1] 12
> var(x1)
[1] 1.734545
> var(x2)
[1] 12.89295
> cov(x1,x2)
[1] -1.070909
> x<-cbind(x1,x2)
> Var(x,diag=T)
       x1        x2
 1.734545 12.892955
> Var(x,diag=F)
          x1        x2
x1  1.734545 -1.070909
x2 -1.070909 12.892955

> k<-2
> x1
[1] 1.2 3.4 5.6 7.5 7.7
> x2
[1] 1.1 2.3 4.4 5.1 2.9
> n<-length(x1)
> n
[1] 5
> var(x1)
[1] 7.717
> var(x2)
[1] 2.588
> cov(x1,x2)
[1] 3.524
> x<-cbind(x1,x2)
```

```
> Var(x,diag=T)
   x1    x2
7.717 2.588
> Var(x,diag=F)
      x1    x2
x1 7.717 3.524
x2 3.524 2.588
```

sd()

- **Package:** `stats`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** deviazione standard
- **Formula:**

$$s_x = \left(\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right)^{1/2}$$

- **Esempio:**

```
> x
[1] 1.0 2.3 4.5 6.7 8.9
> n<-length(x)
> n
[1] 5
> sqrt(sum((x-mean(x))**2)/(n-1))
[1] 3.206556
> sd(x)
[1] 3.206556

> x
[1] 1 3 4 6
> n<-length(x)
> n
[1] 4
> sqrt(sum((x-mean(x))**2)/(n-1))
[1] 2.081666
> sd(x)
[1] 2.081666
```

COV()

- **Package:** `labstatR`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `y` vettore numerico di dimensione n

- **Significato:** covarianza nella popolazione
- **Formula:**

$$\sigma_{xy} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

- **Esempio:**

```
> x
[1] 1.0 2.3 4.5 6.7 8.9
> y
[1] 1 3 4 6 8
> mean((x-mean(x))*(y-mean(y)))
[1] 6.868
> COV(x,y)
[1] 6.868

> x
[1] 1.2 3.4 5.6 7.5 7.7 7.8
> y
[1] 1.1 2.3 4.4 5.1 2.9 8.7
> mean((x-mean(x))*(y-mean(y)))
[1] 4.442222
> COV(x,y)
[1] 4.442222
```

cov()

- **Package:** `stats`
- **Parametri:**
  - `x` vettore numerico di dimensione n
  - `y` vettore numerico di dimensione n
- **Significato:** covarianza campionaria
- **Formula:**

$$s_{xy} = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})$$

- **Esempio:**

```
> x
[1] 1.0 2.3 4.5 6.7 8.9
> y
[1] 1 3 4 6 8
> n<-length(x)
> n
[1] 5
> sum((x-mean(x))*(y-mean(y)))/(n-1)
[1] 8.585
> cov(x,y)
[1] 8.585

> x
[1] 1.0 2.3 4.5 6.7 8.9
> y
[1] 1 3 4 6 8
> n<-length(x)
> n
[1] 5
> sum((x-mean(x))**2)/(n-1)
[1] 10.282
> sum((y-mean(y))**2)/(n-1)
[1] 7.3
> sum((x-mean(x))*(y-mean(y)))/(n-1)
[1] 8.585
> z<-cbind(x,y)
> cov(z)
```

```
       x     y
x 10.282 8.585
y  8.585 7.300
```

cov2cor()

- **Package:** `stats`
- **Parametri:**

  `V` matrice di covarianza di dimensione $k \times k$ relativa ai vettori numerici x_1, x_2, ..., x_k

- **Significato:** converte la matrice di covarianza nella matrice di correlazione
- **Esempio:**

```
> dati
        x   y
[1,] -1.2 1.0
[2,] -1.3 2.0
[3,] -6.7 3.0
[4,]  0.8 5.0
[5,] -7.6 6.0
[6,] -5.6 7.3
> V<-cov(dati)
> V
       x      y
x 12.004 -3.780
y -3.780  5.975
> cor(dati)
           x          y
x  1.0000000 -0.4463339
y -0.4463339  1.0000000
> cov2cor(V)
           x          y
x  1.0000000 -0.4463339
y -0.4463339  1.0000000

> dati
       x    y
[1,] 1.0  2.7
[2,] 2.0 -7.8
[3,] 4.5  8.8
> V<-cov(dati)
> V
      x        y
x 3.250  8.72500
y 8.725 70.50333
> cor(dati)
          x         y
x 1.0000000 0.5763933
y 0.5763933 1.0000000
> cov2cor(V)
          x         y
x 1.0000000 0.5763933
y 0.5763933 1.0000000
```

cov.wt()

- **Package:** `stats`

- **Parametri:**

  `x` matrice di dimensione $n \times k$ le cui colonne corrispondono ai vettori numerici $x_1, x_2, \ldots, x_k$

  `wt` vettore numerico w di pesi a somma unitaria di dimensione n

  `center = T / F` parametro di posizione

  `cor = T / F` correlazione pesata

- **Significato:** matrice di covarianza e correlazione pesata

- **Output:**

  `cov` matrice di covarianza pesata

  `center` media pesata

  `n.obs` dimensione campionaria

  `wt` vettore numerico w

  `cor` matrice di correlazione pesata

- **Formula:**

  `cov`

  `center = T`

  $$s_{x_i x_j} = (1 - w^T w)^{-1} (x_i - \bar{x}_{i\,W})^T \operatorname{diag}(w) (x_j - \bar{x}_{j\,W}) \quad \forall\, i, j = 1, 2, \ldots, k$$

  `center = F`

  $$s_{x_i x_j} = (1 - w^T w)^{-1} x_i^T \operatorname{diag}(w) x_j \quad \forall\, i, j = 1, 2, \ldots, k$$

  `center`

  `center = T`

  $$\bar{x}_{i\,W} = x_i^T w \quad \forall\, i = 1, 2, \ldots, k$$

  `center = F`

  $$0$$

  `n.obs`

  `center = T`

  $$n$$

  `center = F`

  $$n$$

  `wt`

  `center = T`

  $$w$$

  `center = F`

  $$w$$

  `cor`

  `center = T`

  $$r_{x_i x_j} = \frac{(x_i - \bar{x}_{i\,W})^T \operatorname{diag}(w) (x_j - \bar{x}_{j\,W})}{((x_i - \bar{x}_{i\,W})^T \operatorname{diag}(w) (x_i - \bar{x}_{i\,W}))^{1/2} ((x_j - \bar{x}_{j\,W})^T \operatorname{diag}(w) (x_j - \bar{x}_{j\,W}))^{1/2}}$$

  $$\forall\, i, j = 1, 2, \ldots, k$$

  `center = F`

$$r_{x_i x_j} = \frac{x_i^T \operatorname{diag}(w)\, x_j}{\left(x_i^T \operatorname{diag}(w)\, x_i\right)^{1/2} \left(x_j^T \operatorname{diag}(w)\, x_j\right)^{1/2}}$$

$$\forall\, i,\, j = 1, 2, \ldots, k$$

- **Esempio:**

```
> k<-2
> x1
[1] 1.2 3.4 5.6 7.5 7.7 7.8
> x2
[1] 1.1 2.3 4.4 5.1 2.9 8.7
> n<-length(x1)
> n
[1] 6
> prova<-abs(rnorm(6))
> pesi<-prova/sum(prova)
> sum(pesi)
[1] 1
> x1W<-sum(x1*pesi)
> x2W<-sum(x2*pesi)
> as.numeric(1/(1-t(pesi)%*%pesi)*t(x1-x1W)%*%diag(pesi)%*%(x1-x1W))
[1] 8.002225
> as.numeric(1/(1-t(pesi)%*%pesi)*t(x2-x2W)%*%diag(pesi)%*%(x2-x2W))
[1] 7.783884
> as.numeric(1/(1-t(pesi)%*%pesi)*t(x1-x1W)%*%diag(pesi)%*%(x2-x2W))
[1] 6.56038
> z<-cbind(x1,x2)
> cov.wt(z,wt=pesi,center=T,cor=T)$cov
         x1       x2
x1 8.002225 6.560380
x2 6.560380 7.783884
> as.numeric(1/(1-t(pesi)%*%pesi)*t(x1)%*%diag(pesi)%*%x1)
[1] 37.23229
> as.numeric(1/(1-t(pesi)%*%pesi)*t(x2)%*%diag(pesi)%*%x2)
[1] 26.33601
> as.numeric(1/(1-t(pesi)%*%pesi)*t(x1)%*%diag(pesi)%*%x2)
[1] 29.84728
> cov.wt(z,wt=pesi,center=F,cor=T)$cov
         x1       x2
x1 37.23229 29.84728
x2 29.84728 26.33601
> c(x1W,x2W)
[1] 4.854610 3.867553
> cov.wt(z,wt=pesi,center=T,cor=T)$center
      x1       x2
4.854610 3.867553
> cov.wt(z,wt=pesi,center=F,cor=T)$center
[1] 0
> n
[1] 6
> cov.wt(z,wt=pesi,center=T,cor=T)$n.obs
[1] 6
> cov.wt(z,wt=pesi,center=F,cor=T)$n.obs
[1] 6
> pesi
[1] 0.245156 0.16509 0.25690 0.09221 0.08210 0.15856
> cov.wt(z,wt=pesi,center=T,cor=T)$wt
[1] 0.245156 0.16509 0.25690 0.09221 0.08210 0.15856
> cov.wt(z,wt=pesi,center=F,cor=T)$wt
[1] 0.245156 0.16509 0.25690 0.09221 0.08210 0.15856
> covx1x2<-1/(1-t(pesi)%*%pesi)*t(x1-x1W)%*%diag(pesi)%*%(x2-x2W)
> covx1x2<-as.numeric(covx1x2)
```

```
> covx1x2
[1] 6.56038
> sx1<-sqrt(1/(1-t(pesi)%*%pesi)*t(x1-x1W)%*%diag(pesi)%*%(x1-x1W))
> sx1<-as.numeric(sx1)
> sx1
[1] 2.828820
> sx2<-sqrt(1/(1-t(pesi)%*%pesi)*t(x2-x2W)%*%diag(pesi)%*%(x2-x2W))
> sx2<-as.numeric(sx2)
> sx2
[1] 2.789961
> rx1x2<-covx1x2/(sx1*sx2)
> rx1x2
[1] 0.831238
> cov.wt(z,wt=pesi,center=T,cor=T)$cor
          [,1]     [,2]
[1,] 1.000000 0.831238
[2,] 0.831238 1.000000
> covx1x2<-as.numeric(1/(1-t(pesi)%*%pesi)*t(x1)%*%diag(pesi)%*%x2)
> covx1x2
[1] 29.84728
> sx1<-sqrt(as.numeric(1/(1-t(pesi)%*%pesi)*t(x1)%*%diag(pesi)%*%x1))
> sx1
[1] 6.101826
> sx2<-sqrt(as.numeric(1/(1-t(pesi)%*%pesi)*t(x2)%*%diag(pesi)%*%x2))
> sx2
[1] 5.131862
> rx1x2<-covx1x2/(sx1*sx2)
> rx1x2
[1] 0.953169
> cov.wt(z,wt=pesi,center=F,cor=T)$cor
          [,1]     [,2]
[1,] 1.000000 0.953169
[2,] 0.953169 1.000000
```

3.4 Indici di forma

skew()

- **Package:** `labstatR`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** asimmetria nella popolazione
- **Formula:**

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i-\bar{x}}{\sigma_x}\right)^3$$

- **Esempio:**

```
> x
[1] 1.0 2.3 4.5 6.7 8.9
> sigmax<-sqrt(sigma2(x))
> mean((x-mean(x))**3/sigmax**3)
[1] 0.1701538
> skew(x)
[1] 0.1701538

> x
[1] 1.2 3.4 5.2 3.4 4.4
```

```
> sigmax<-sqrt(sigma2(x))
> mean((x-mean(x))**3/sigmax**3)
[1] -0.5845336
> skew(x)
[1] -0.5845336
```

skewness()

- **Package:** `fBasics`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** asimmetria campionaria
- **Formula:**

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i-\bar{x}}{s_x}\right)^3$$

- **Esempio:**

```
> x
[1] 1.0 2.3 4.5 6.7 8.9
> mean((x-mean(x))**3/sd(x)**3)
[1] 0.1217521
> skewness(x)
[1] 0.1217521

> x
[1] 1.2 3.4 5.2 3.4 4.4
> mean((x-mean(x))**3/sd(x)**3)
[1] -0.4182582
> skewness(x)
[1] -0.4182582
```

kurt()

- **Package:** `labstatR`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** kurtosi nella popolazione
- **Formula:**

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i-\bar{x}}{\sigma_x}\right)^4$$

- **Esempio:**

```
> x
[1] 1.0 2.3 4.5 6.7 8.9
> sigmax<-sqrt(sigma2(x))
> mean((x-mean(x))**4/sigmax**4)
[1] 1.623612
> kurt(x)
[1] 1.623612

> x
[1] 1.2 3.4 5.2 3.4 4.4
```

```
> sigmax<-sqrt(sigma2(x))
> mean((x-mean(x))**4/sigmax**4)
[1] 2.312941
> kurt(x)
[1] 2.312941
```

kurtosis()

- **Package:** `fBasics`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** kurtosi campionaria
- **Formula:**

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{x_i-\bar{x}}{s_x}\right)^4-3$$

- **Esempio:**

```
> x
[1] 1.0 2.3 4.5 6.7 8.9
> mean((x-mean(x))**4/sd(x)**4)-3
[1] -1.960889
> kurtosis(x)
[1] -1.960889

> x
[1] 1.2 3.4 5.2 3.4 4.4
> mean((x-mean(x))**4/sd(x)**4)-3
[1] -1.519718
> kurtosis(x)
[1] -1.519718
```

geary()

- **Package:** `moments`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** kurtosi secondo *Geary*
- **Formula:**

$$\frac{\sum_{i=1}^{n}|\,x_i-\bar{x}\,|}{n\,\sigma_x}$$

- **Esempio:**

```
> x
[1] 1.0 2.3 4.5 6.7 8.9
> sx<-sqrt(mean((x-mean(x))**2))
> sum(abs(x-mean(x)))/(length(x)*sx)
[1] 0.8702836
> geary(x)
[1] 0.8702836

> x
[1] 1.2 3.4 5.2 3.4 4.4
> sx<-sqrt(mean((x-mean(x))**2))
```

```
> sum(abs(x-mean(x)))/(length(x)*sx)
[1] 0.7629055
> geary(x)
[1] 0.7629055
```

3.5 Indici di correlazione

cor()

- **Package:** `stats`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `y` vettore numerico di dimensione n

  `method = pearson / spearman / kendall` tipo di coefficiente

- **Significato:** coefficiente di correlazione lineare
- **Formula:**

`method = pearson`

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\left(\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{1/2}\left(\sum_{i=1}^{n}(y_i - \bar{y})^2\right)^{1/2}}$$

`method = spearman`

$$r_{xy}^{S} = \frac{\sum_{i=1}^{n}(a_i - \bar{a})(b_i - \bar{b})}{\left(\sum_{i=1}^{n}(a_i - \bar{a})^2\right)^{1/2}\left(\sum_{i=1}^{n}(b_i - \bar{b})^2\right)^{1/2}}$$

dove a, b sono i ranghi di x ed y rispettivamente.

`method = kendall`

$$r_{xy}^{K} = \frac{2\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\operatorname{sign}((x_j - x_i)(y_j - y_i))}{\left(n(n-1) - \sum_{i=1}^{g} t_i(t_i - 1)\right)^{1/2}\left(n(n-1) - \sum_{j=1}^{h} u_j(u_j - 1)\right)^{1/2}}$$

dove t, u sono i ties di x ed y rispettivamente.

- **Esempio:**

```
> # coefficiente di pearson
> x
[1] 1 2 2 4 3 3
> y
[1] 6 6 7 7 7 9
> cov(x,y)/(sd(x)*sd(y))
[1] 0.522233
> cor(x,y,method="pearson")
[1] 0.522233

> # coefficiente di pearson
> x
[1] 1.0 2.0 3.0 5.6 7.6 2.3 1.0
> y
```

```
[1] 1.2  2.2  3.0 15.6 71.6  2.2  1.2
> cov(x,y)/(sd(x)*sd(y))
[1] 0.8790885
> cor(x,y,method="pearson")
[1] 0.8790885

> # coefficiente di spearman
> x
[1] 1 2 2 4 3 3
> y
[1] 6 6 7 7 7 9
> a<-rank(x)
> b<-rank(y)
> cov(a,b)/(sd(a)*sd(b))
[1] 0.6833149
> cor(x,y,method="spearman")
[1] 0.6833149

> # coefficiente di spearman
> x
[1] 1.0 2.0 3.0 5.6 7.6 2.3 1.0
> y
[1] 1.2  2.2  3.0 15.6 71.6  2.2  1.2
> a<-rank(x)
> b<-rank(y)
> cov(a,b)/(sd(a)*sd(b))
[1] 0.9908674
> cor(a,b,method="pearson")
[1] 0.9908674

> # coefficiente di kendall
> x
[1] 1 2 2 4 3 3
> y
[1] 6 6 7 7 7 9
> n<-length(x)
> n
[1] 6
> matrice<-matrix(0,nrow=n-1,ncol=n,byrow=F)
> for(i in 1:(n-1))
+ for(j in (i+1):n)
+ matrice[i,j]<-sign((x[j]-x[i])*(y[j]-y[i]))
> num<-2*sum(matrice)
> table(rank(x))

  1 2.5 4.5   6
  1   2   2   1
> g<-2
> t1<-2
> t2<-2
> t<-c(t1,t2)
> t
[1] 2 2
> table(rank(y))

1.5   4   6
  2   3   1
> h<-2
> u1<-2
> u2<-3
> u<-c(u1,u2)
> u
[1] 2 3
```

```
> den<-(n*(n-1)-sum(t*(t-1)))**0.5*(n*(n-1)-sum(u*(u-1)))**0.5
> num/den
[1] 0.5853694
> cor(x,y,method="kendall")
[1] 0.5853694

> # coefficiente di kendall
> x
[1] 1.0 2.0 3.0 5.6 7.6 2.3 1.0
> y
[1] 1.2  2.2  3.0 15.6 71.6  2.2  1.2
> n<-length(x)
> n
[1] 7
> matrice<-matrix(0,nrow=n-1,ncol=n,byrow=F)
> for(i in 1:(n-1))
+ for(j in (i+1):n)
+ matrice[i,j]<-sign((x[j]-x[i])*(y[j]-y[i]))
> num<-2*sum(matrice)
> table(rank(x))

1.5   3   4   5   6   7
  2   1   1   1   1   1
> g<-1
> t<-2
> table(rank(y))

1.5 3.5   5   6   7
  2   2   1   1   1
> h<-2
> u1<-2
> u2<-2
> u<-c(u1,u2)
> u
[1] 2 2
> den<-(n*(n-1)-sum(t*(t-1)))**0.5*(n*(n-1)-sum(u*(u-1)))**0.5
> num/den
[1] 0.9746794
> cor(x,y,method="kendall")
[1] 0.9746794
```

cancor()

- **Package:** `stats`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `y` vettore numerico di dimensione n

  `xcenter = T / F` parametro di posizione

  `ycenter = T / F` parametro di posizione

- **Significato:** correlazione lineare canonica
- **Output:**

  `cor` coefficiente di correlazione lineare

  `xcenter` parametro di locazione

  `ycenter` parametro di locazione

- **Formula:**

	ycenter = T	ycenter = F
xcenter = T	$\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\left(\sum_{i=1}^{n}(x_i-\bar{x})^2\right)^{1/2}\left(\sum_{i=1}^{n}(y_i-\bar{y})^2\right)^{1/2}}$	$\frac{\sum_{i=1}^{n}(x_i-\bar{x})y_i}{\left(\sum_{i=1}^{n}(x_i-\bar{x})^2\right)^{1/2}\left(\sum_{i=1}^{n}y_i^2\right)^{1/2}}$
xcenter = F	$\frac{\sum_{i=1}^{n}x_i(y_i-\bar{y})}{\left(\sum_{i=1}^{n}x_i^2\right)^{1/2}\left(\sum_{i=1}^{n}(y_i-\bar{y})^2\right)^{1/2}}$	$\frac{\sum_{i=1}^{n}x_i y_i}{\left(\sum_{i=1}^{n}x_i^2\right)^{1/2}\left(\sum_{i=1}^{n}y_i^2\right)^{1/2}}$

	ycenter = T	ycenter = F
xcenter = T	$\bar{x}$	$\bar{x}$
xcenter = F	0	0

cor

xcenter

ycenter

	ycenter = T	ycenter = F
xcenter = T	$\bar{y}$	0
xcenter = F	$\bar{y}$	0

- **Esempio:**

```
> x
[1] 1.0 2.0 3.0 5.6 7.6 2.3 1.0
> y
[1] 1.2  2.2  3.0 15.6 71.6  2.2  1.2
> n<-length(x)
> n
[1] 7
> sum((x-mean(x))*(y-mean(y)))/(sum((x-mean(x))**2)**0.5*sum((y-mean(y))**2)**0.5)
[1] 0.8790885
> cancor(x,y,xcenter=T,ycenter=T)$cor
[1] 0.8790885
> mean(x)
[1] 3.214286
> cancor(x,y,xcenter=T,ycenter=T)$xcenter
[1] 3.214286
> mean(y)
[1] 13.85714
> cancor(x,y,xcenter=T,ycenter=T)$ycenter
[1] 13.85714
> sum((x-mean(x))*y)/(sum((x-mean(x))**2)**0.5*sum(y**2)**0.5)
[1] 0.7616638
> cancor(x,y,xcenter=T,ycenter=F)$cor
[1] 0.7616638
> mean(x)
[1] 3.214286
> cancor(x,y,xcenter=T,ycenter=F)$xcenter
[1] 3.214286
> cancor(x,y,xcenter=T,ycenter=F)$ycenter
[1] 0
> sum(x*(y-mean(y)))/(sum(x**2)**0.5*sum((y-mean(y))**2)**0.5)
[1] 0.5118281
> cancor(x,y,xcenter=F,ycenter=T)$cor
[1] 0.5118281
> cancor(x,y,xcenter=F,ycenter=T)$xcenter
[1] 0
> mean(y)
[1] 13.85714
> cancor(x,y,xcenter=F,ycenter=T)$ycenter
[1] 13.85714
> sum(x*y)/(sum(x**2)**0.5*sum(y**2)**0.5)
[1] 0.8494115
```

```
> cancor(x,y,xcenter=F,ycenter=F)$cor
[1] 0.8494115
> cancor(x,y,xcenter=F,ycenter=F)$xcenter
[1] 0
> cancor(x,y,xcenter=F,ycenter=F)$ycenter
[1] 0
```

partial.cor()

- **Package:** `Rcmdr`
- **Parametri:**

  `X` matrice di dimensione $n \times k$ le cui colonne corrispondono ai vettori numerici x_1, x_2, ..., x_k
- **Significato:** correlazione parziale
- **Formula:**

$$r_{x_i x_j|\cdot} = -\frac{R^{-1}_{i,j}}{\sqrt{R^{-1}_{i,i} R^{-1}_{j,j}}} \quad \forall i \neq j = 1, 2, \ldots, k$$

  dove R è la matrice di correlazione tra i k vettori
- **Esempio:**

```
> n<-nrow(X)
> k<-ncol(X)
> R<-cor(X)
> partial.cor(X)
```

corr()

- **Package:** `boot`
- **Parametri:**

  `d` matrice di dimensione $n \times 2$ le cui colonne corrispondono ai vettori numerici x ed y

  `w` vettore numerico w di pesi a somma unitaria di dimensione n
- **Significato:** correlazione pesata
- **Formula:**

$$r_{xy} = \frac{(x - \bar{x}_W)^T \operatorname{diag}(w) (y - \bar{y}_W)}{\left((x - \bar{x}_W)^T \operatorname{diag}(w) (x - \bar{x}_W)\right)^{1/2} \left((y - \bar{y}_W)^T \operatorname{diag}(w) (y - \bar{y}_W)\right)^{1/2}}$$

- **Esempio:**

```
> x
[1] 1.2 2.3 3.4 4.5 5.6 6.7
> y
[1] 1.0 2.0 3.0 5.0 6.0 7.3
> d<-as.matrix(cbind(x,y))
> n<-nrow(d)
> w<-abs(rnorm(n))
> w<-w/sum(w)
> sum(w)
[1] 1
> mxw<-weighted.mean(x,w)
> myw<-weighted.mean(y,w)
> num<-as.numeric(t(x-mxw)%*%diag(w)%*%(y-myw))
> den<-as.numeric(sqrt(t(x-mxw)%*%diag(w)%*%(x-mxw)*t(y-myw)%*%diag(w)%*%(y-myw)))
```

```
> coefcorr<-num/den
> coefcorr
[1] 0.9971414
> corr(d,w)
[1] 0.9971414

> x
[1] 1.0 2.0 3.0 5.6 7.6 2.3 1.0
> y
[1]  1.2  2.2  3.0 15.6 71.6  2.2  1.2
> d<-as.matrix(cbind(x,y))
> n<-nrow(d)
> w<-abs(rnorm(n))
> w<-w/sum(w)
> sum(w)
[1] 1
> mxw<-weighted.mean(x,w)
> myw<-weighted.mean(y,w)
> num<-as.numeric(t(x-mxw)%*%diag(w)%*%(y-myw))
> den<-as.numeric(sqrt(t(x-mxw)%*%diag(w)%*%(x-mxw)*t(y-myw)%*%diag(w)%*%(y-myw)))
> coefcorr<-num/den
> coefcorr
[1] 0.9566433
> corr(d,w)
[1] 0.9566433
```

acf()

- **Package:** `stats`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `lag.max` il valore d del ritardo

  `type = correlation / covariance / partial` tipo di legame

  `demean = T / F` centratura

- **Significato:** autocovarianza oppure autocorrelazione
- **Output:**

  `acf` autocovarianza oppure autocorrelazione

  `n.used` dimensione campionaria

  `lag` il valore d del ritardo

- **Formula:**

  `acf`

`type = correlation` AND `demean = T`

$$\hat{\rho}(k) = \frac{\sum_{t=1}^{n-k} (x_t - \bar{x})\,(x_{t+k} - \bar{x})}{\sum_{t=1}^{n} (x_t - \bar{x})^2} \quad \forall\, k = 0,\, 1,\, 2,\, \ldots,\, d$$

`type = correlation` AND `demean = F`

$$\hat{\rho}(k) = \frac{\sum_{t=1}^{n-k} x_t\, x_{t+k}}{\sum_{t=1}^{n} x_t^2} \quad \forall\, k = 0,\, 1,\, 2,\, \ldots,\, d$$

`type = covariance` AND `demean = T`

$$\hat{\gamma}(k) = \frac{1}{n} \sum_{t=1}^{n-k} (x_t - \bar{x})\,(x_{t+k} - \bar{x}) \quad \forall\, k = 0, 1, 2, \ldots, d$$

`type = covariance` AND `demean = F`

$$\hat{\gamma}(k) = \frac{1}{n} \sum_{t=1}^{n-k} x_t\, x_{t+k} \quad \forall\, k = 0, 1, 2, \ldots, d$$

`type = partial` AND `demean = T / F`

$$\hat{\pi}(k) = \frac{\begin{vmatrix} 1 & \hat{\rho}(1) & \hat{\rho}(2) & \ldots & \hat{\rho}(1) \\ \hat{\rho}(1) & 1 & \hat{\rho}(1) & \ldots & \hat{\rho}(2) \\ \hat{\rho}(2) & \hat{\rho}(1) & 1 & \ldots & \hat{\rho}(3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{\rho}(k-1) & \hat{\rho}(k-2) & \hat{\rho}(k-3) & \ldots & \hat{\rho}(k) \end{vmatrix}}{\begin{vmatrix} 1 & \hat{\rho}(1) & \hat{\rho}(2) & \ldots & \hat{\rho}(k-1) \\ \hat{\rho}(1) & 1 & \hat{\rho}(1) & \ldots & \hat{\rho}(k-2) \\ \hat{\rho}(2) & \hat{\rho}(1) & 1 & \ldots & \hat{\rho}(k-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{\rho}(k-1) & \hat{\rho}(k-2) & \hat{\rho}(k-3) & \ldots & 1 \end{vmatrix}} \quad \forall\, k = 1, 2, \ldots, d$$

`n.used`

n

`lag`

d

- **Esempio:**

```
> x
[1] 1 2 7 3 5 2 0 1 4 5
> n<-length(x)
> n
[1] 10
> d<-4
> sum((x[1:(n-d)]-mean(x))*(x[(d+1):n]-mean(x)))/((n-1)*var(x))
[1] -0.3409091
> acf(x,lag=d,type="correlation",demean=T,plot=F)$acf[d+1]
[1] -0.3409091

> x
[1] 1 2 7 3 5 2 0 1 4 5
> n<-length(x)
> n
[1] 10
> d<-4
> sum((x[1:(n-d)]-mean(x))*(x[(d+1):n]-mean(x)))/n
[1] -1.5
> acf(x,lag=d,type="covariance",demean=T,plot=F)$acf[d+1]
[1] -1.5
```

pacf()

- **Package:** `stats`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `lag.max` il valore d del ritardo

  `demean = T / F` centratura

- **Significato:** autocorrelazione parziale
- **Output:**

  `acf` autocorrelazione parziale

  `n.used` dimensione campionaria

  `lag` il valore d del ritardo

- **Formula:**

  `acf`

$$\hat{\pi}(k) = \frac{\begin{vmatrix} 1 & \hat{\rho}(1) & \hat{\rho}(2) & \dots & \hat{\rho}(1) \\ \hat{\rho}(1) & 1 & \hat{\rho}(1) & \dots & \hat{\rho}(2) \\ \hat{\rho}(2) & \hat{\rho}(1) & 1 & \dots & \hat{\rho}(3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{\rho}(k-1) & \hat{\rho}(k-2) & \hat{\rho}(k-3) & \dots & \hat{\rho}(k) \end{vmatrix}}{\begin{vmatrix} 1 & \hat{\rho}(1) & \hat{\rho}(2) & \dots & \hat{\rho}(k-1) \\ \hat{\rho}(1) & 1 & \hat{\rho}(1) & \dots & \hat{\rho}(k-2) \\ \hat{\rho}(2) & \hat{\rho}(1) & 1 & \dots & \hat{\rho}(k-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{\rho}(k-1) & \hat{\rho}(k-2) & \hat{\rho}(k-3) & \dots & 1 \end{vmatrix}} \quad \forall\, k = 1, 2, \dots, d$$

`demean = T`

$$\hat{\rho}(k) = \frac{\sum_{t=1}^{n-k} (x_t - \bar{x})\,(x_{t+k} - \bar{x})}{\sum_{t=1}^{n} (x_t - \bar{x})^2} \quad \forall\, k = 0, 1, 2, \dots, d$$

`demean = F`

$$\hat{\rho}(k) = \frac{\sum_{t=1}^{n-k} x_t\, x_{t+k}}{\sum_{t=1}^{n} x_t^2} \quad \forall\, k = 0, 1, 2, \dots, d$$

`n.used`

$$n$$

`lag`

$$d$$

- **Esempio:**

```
> pacf(x,lag=d,demean=T,plot=F)
```

3.6 Indici di connessione e di dipendenza in media

eta()

- **Package:** `labstatR`
- **Parametri:**

  `y` vettore numerico di dimensione n

  `f` fattore a k livelli di dimensione n

- **Significato:** $\eta^2_{y|f}$
- **Formula:**

$$\eta^2_{y|f} = \frac{\sum_{j=1}^{k} (\bar{y}_j - \bar{y})^2\, n_j}{\sum_{i=1}^{n} (\bar{y}_i - \bar{y})^2}$$

- **Esempio:**

```
> y
[1] 1.0 1.2 2.1 3.4 5.4 5.6 7.2 3.2 3.0 1.0 2.3
> f
[1] a b c b a c a b b c a
Levels: a b c
> k<-nlevels(f)
> k
[1] 3
> n<-length(f)
> n
[1] 11
> table(f)
f
a b c
4 4 3
> n1<-4
> n2<-4
> n3<-3
> enne<-c(n1,n2,n3)
> enne
[1] 4 4 3
> y1medio<-mean(y[f=="a"])
> y2medio<-mean(y[f=="b"])
> y3medio<-mean(y[f=="c"])
> ymedio<-c(y1medio,y2medio,y3medio)
> ymedio
[1] 3.975 2.700 2.900
> sum((ymedio-mean(y))**2*enne)/sum((y-mean(y))**2)
[1] 0.08657807
> eta(f,y)
[1] 0.08657807
```

gini()

- **Package:** `labstatR`
- **Parametri:**

  `y` vettore numerico di dimensione n

- **Significato:** indici di concentrazione
- **Output:**

  `G` indice di *Gini*

  `R` rapporto di concentrazione di *Gini*

  `P` proporzioni

  `Q` somme cumulate

- **Formula:**

  `G`

$$G = \frac{2}{n-1} \sum_{i=1}^{n} \left(\frac{i}{n} - \frac{\sum_{j=1}^{i} y_{(j)}}{\sum_{j=1}^{n} y_j} \right)$$

  `R`

$$\frac{n-1}{n} G$$

  `P`

$$0, i/n \quad \forall i = 1, 2, \ldots, n$$

`Q`

$$0, \sum_{j=1}^{i} y_{(j)} \Big/ \sum_{j=1}^{n} y_j \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> y<-c(1,1,1,4,4,5,7,10)
> y<-sort(y,decreasing=F)
> n<-length(y)
> G<-2/(n-1)*sum((1:n)/n-cumsum(y)/sum(y))
> G
[1] 0.455
> gini(y,plot=F)$G
[1] 0.455
> R<-(n-1)/n*G
> R
[1] 0.398
> gini(y,plot=F)$R
[1] 0.398
> P<-c(0,(1:n)/n)
> P
[1] 0.000 0.125 0.250 0.375 0.500 0.625 0.750 0.875 1.000
> gini(y,plot=F)$P
[1] 0.000 0.125 0.250 0.375 0.500 0.625 0.750 0.875 1.000
> Q<-c(0,cumsum(y)/sum(y))
> Q
[1] 0.0000 0.0303 0.0606 0.0909 0.2121 0.3333 0.4848 0.6970 1.0000
> gini(y,plot=F)$Q
[1] 0.0000 0.0303 0.0606 0.0909 0.2121 0.3333 0.4848 0.6970 1.0000
```

chi2()

- **Package:** `labstatR`
- **Parametri:**

  `f` fattore a k livelli

  `g` fattore a h livelli

- **Significato:** indice di connessione $\tilde{\chi}^2$
- **Formula:**

$$\tilde{\chi}^2 = \frac{\chi^2}{\chi^2_{\max}} = \frac{\sum_{i=1}^{k} \sum_{j=1}^{h} \frac{(n_{ij} - \hat{n}_{ij})^2}{\hat{n}_{ij}}}{n_{..} \min(k-1,\, h-1)} = \frac{\sum_{i=1}^{k} \sum_{j=1}^{h} \frac{n_{ij}^2}{n_{i\cdot}\, n_{\cdot j}} - 1}{\min(k-1,\, h-1)}$$

$$\text{dove} \qquad \hat{n}_{ij} = \frac{n_{ij}}{n_{i\cdot}\, n_{\cdot j}} \quad \forall\, i = 1, 2, \ldots, k \quad \forall\, j = 1, 2, \ldots, h$$

$$n_{..} = \sum_{i=1}^{k} \sum_{j=1}^{h} n_{ij}$$

- **Esempio:**

```
> f
[1] a b c b a c a b b c a
Levels: a b c
> k<-nlevels(f)
> k
[1] 3
> g
[1] O P W P P O O W W P P
Levels: O P W
```

```
> h<-nlevels(g)
> h
[1] 3
> table(f,g)
   g
f   O P W
  a 2 2 0
  b 0 2 2
  c 1 1 1
> n..<-sum(table(f,g))
> n..
[1] 11
> chi2(f,g)
[1] 0.1777778
```

E()

- **Package:** `labstatR`
- **Parametri:**

  `f` fattore a k livelli di dimensione n

- **Significato:** indice di eterogeneità di *Gini*
- **Formula:**

$$E = \frac{k}{k-1}\left(1 - \frac{1}{n^2}\sum_{i=1}^{k} n_i^2\right)$$

- **Esempio:**

```
> f
[1] a b c b a c a b b c a
Levels: a b c
> n<-length(f)
> n
[1] 11
> k<-nlevels(f)
> k
[1] 3
> table(f)
f
a b c
4 4 3
> n1<-4
> n2<-4
> n3<-3
> enne<-c(n1,n2,n3)
> enne
[1] 4 4 3
> E<-k/(k-1)*(1-1/n**2*sum(enne**2))
> E
[1] 0.9917355
> E(f)
[1] 0.9917355
```

3.7 Funzioni di sintesi

summary()

- **Package:** `base`

- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** minimo, primo quartile, mediana, media, terzo quartile e massimo

- **Formula:**

$$x_{(1)} \qquad Q_{0.25}(x) \qquad Q_{0.5}(x) \qquad \bar{x} \qquad Q_{0.75}(x) \qquad x_{(n)}$$

- **Esempio:**

```
> x
[1] 1.0 2.3 5.0 6.7 8.0
> min(x)
[1] 1
> quantile(x,probs=0.25)
25%
2.3
> median(x)
[1] 5
> mean(x)
[1] 4.6
> quantile(x,probs=0.75)
75%
6.7
> max(x)
[1] 8
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0     2.3     5.0     4.6     6.7     8.0

> x
[1]  1.2  2.2  3.0 15.6 71.6  2.2  1.2
> min(x)
[1] 1.2
> quantile(x,probs=0.25)
25%
1.7
> median(x)
[1] 2.2
> mean(x)
[1] 13.85714
> quantile(x,probs=0.75)
75%
9.3
> max(x)
[1] 71.6
> summary(x)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.20    1.70    2.20   13.86    9.30   71.60
```

- **Osservazioni:** Calcola i quartili con la funzione `quantile()`.

fivenum()

- **Package:** `stats`

- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** cinque numeri di *Tukey*

- **Formula:**

$$x_{(1)} \qquad Q_{0.5}\left(x_i\,|_{x_i \leq Q_{0.5}(x)}\right) \qquad Q_{0.5}(x) \qquad Q_{0.5}\left(x_i\,|_{x_i \geq Q_{0.5}(x)}\right) \qquad x_{(n)}$$

- **Esempio:**

```
> x
[1] 1.0 2.3 5.0 6.7 8.0
> min(x)
[1] 1
> median(x[x<=median(x)])
[1] 2.3
> median(x)
[1] 5
> median(x[x>=median(x)])
[1] 6.7
> max(x)
[1] 8
> fivenum(x)
[1] 1.0 2.3 5.0 6.7 8.0

> x
[1]  1.2  1.2  2.2  2.2  3.0 15.6 71.6
> median(x)
[1] 2.2
> x<-x[-3]
> x
[1]  1.2  1.2  2.2  3.0 15.6 71.6
> min(x)
[1] 1.2
> median(x[x<=2.2])
[1] 1.2
> median(x)
[1] 2.6
> median(x[x>=2.2])
[1] 9.3
> max(x)
[1] 71.6
> fivenum(x)
[1]  1.2  1.2  2.6 15.6 71.6
```

- **Osservazioni:** Se x contiene k volte ($k > 1$) il valore $Q_{0.5}(x)$, occorre considerare un nuovo vettore x con sole $k - 1$ replicazioni.

basicStats()

- **Package:** `fBasics`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `ci` livello di confidenza $1 - \alpha$

- **Significato:** statistiche riassuntive
- **Output:**

  dimensione campionaria

  numero di valori `NA` oppure `NaN`

  minimo

  massimo

  primo quartile

  terzo quartile

media aritmetica

mediana

somma

errore standard della media

estremo inferiore dell'intervallo di confidenza a livello $1 - \alpha$ per la media incognita

estremo superiore dell'intervallo di confidenza a livello $1 - \alpha$ per la media incognita

varianza campionaria

deviazione standard

asimmetria campionaria

kurtosi campionaria

- **Formula:**

$$n$$

$$\#\,\texttt{NA} + \#\,\texttt{NaN}$$

$$x_{(1)}$$

$$x_{(n)}$$

$$Q_{0.25}(x)$$

$$Q_{0.75}(x)$$

$$\bar{x}$$

$$Q_{0.5}(x)$$

$$\sum_{i=1}^{n} x_i$$

$$s_x \,/\, \sqrt{n}$$

$$\bar{x} - t_{1-\alpha\,/\,2,\,n-1}\, s_x \,/\, \sqrt{n}$$

$$\bar{x} + t_{1-\alpha\,/\,2,\,n-1}\, s_x \,/\, \sqrt{n}$$

$$s_x^2$$

$$s_x$$

$$\frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x} \right)^3$$

$$\frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x} \right)^4 - 3$$

- **Esempio:**

```
> x
[1] 1.0 2.3 5.0 6.7 8.0
> length(x)
[1] 5
> sum(is.na(x))
[1] 0
> min(x)
[1] 1
> max(x)
[1] 8
> quantile(x,probs=0.25)
25%
2.3
> quantile(x,probs=0.75)
75%
6.7
> mean(x)
[1] 4.6
> median(x)
[1] 5
> sum(x)
[1] 23
> sd(x)/sqrt(length(x))
[1] 1.311106
> alpha<-0.05
> mean(x)-qt(1-alpha/2,length(x)-1)*sd(x)/sqrt(length(x))
[1] 0.959785
> mean(x)+qt(1-alpha/2,length(x)-1)*sd(x)/sqrt(length(x))
[1] 8.240215
> var(x)
[1] 8.595
> sd(x)
[1] 2.931723
> mean((x-mean(x))**3/sd(x)**3)
[1] -0.08091067
> mean((x-mean(x))**4/sd(x)**4)-3
[1] -2.055005
> basicStats(x)
                 Value
nobs         5.00000000
NAs          0.00000000
Minimum      1.00000000
Maximum      8.00000000
1. Quartile  2.30000000
3. Quartile  6.70000000
Mean         4.60000000
Median       5.00000000
Sum         23.00000000
SE Mean      1.31110640
LCL Mean     0.95978504
UCL Mean     8.24021496
Variance     8.59500000
Stdev        2.93172304
Skewness    -0.08091067
Kurtosis    -2.05500479

> x
[1]  1.2  1.2  2.2  3.0 15.6 71.6
> length(x)
[1] 6
> sum(is.na(x))
[1] 0
> min(x)
```

```
[1] 1.2
> max(x)
[1] 71.6
> quantile(x,probs=0.25)
 25%
1.45
> quantile(x,probs=0.75)
  75%
12.45
> mean(x)
[1] 15.8
> median(x)
[1] 2.6
> sum(x)
[1] 94.8
> sd(x)/sqrt(length(x))
[1] 11.38537
> alpha<-0.05
> mean(x)-qt(1-alpha/2,length(x)-1)*sd(x)/sqrt(length(x))
[1] -13.46703
> mean(x)+qt(1-alpha/2,length(x)-1)*sd(x)/sqrt(length(x))
[1] 45.06703
> var(x)
[1] 777.76
> sd(x)
[1] 27.88835
> mean((x-mean(x))**3/sd(x)**3)
[1] 1.251736
> mean((x-mean(x))**4/sd(x)**4)-3
[1] -0.2870146
> basicStats(x)
                  Value
nobs          6.0000000
NAs           0.0000000
Minimum       1.2000000
Maximum      71.6000000
1. Quartile   1.4500000
3. Quartile  12.4500000
Mean         15.8000000
Median        2.6000000
Sum          94.8000000
SE Mean      11.3853707
LCL Mean    -13.4670272
UCL Mean     45.0670272
Variance    777.7600000
Stdev        27.8883488
Skewness      1.2517358
Kurtosis     -0.2870146
```

- **Osservazioni:** Calcola i quartili con la funzione `quantile()`.

boxplot.stats()

- **Package:** `grDevices`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `coef` valore c positivo

- **Significato:** statistiche necessarie per il boxplot
- **Output:**

`stats` cinque numeri di *Tukey*
`n` dimensione del vettore x
`conf` intervallo di *notch*
`out` valori di x esterni all'intervallo tra i *baffi*

- **Formula:**

  stats
  $$x_{(1)} \qquad Q_{0.5}\left(x_i\,|_{x_i \le Q_{0.5}(x)}\right) \qquad Q_{0.5}(x) \qquad Q_{0.5}\left(x_i\,|_{x_i \ge Q_{0.5}(x)}\right) \qquad x_{(n)}$$
  n
  $$n$$
  conf
  $$Q_{0.5}(x) \mp 1.58 \cdot IQR(x) \,/\, \sqrt{n}$$
  out
  $$x_i < Q_{0.25}(x) - c \cdot IQR(x) \quad OR \quad x_i > Q_{0.75}(x) + c \cdot IQR(x)$$

- **Esempio:**

```
> x
[1]  1.2  1.2  2.2  3.0 15.6 71.6
> c<-1.4
> fn<-fivenum(x)
> fn
[1]  1.2  1.2  2.6 15.6 71.6
> boxplot.stats(x,coef=c)$stats
[1]  1.2  1.2  2.6 15.6 15.6
> n<-length(x)
> n
[1] 6
> boxplot.stats(x,coef=c)$n
[1] 6
> median(x)+c(-1,1)*1.58*(fn[4]-fn[2])/sqrt(n)
[1] -6.688465 11.888465
> boxplot.stats(x,coef=c)$conf
[1] -6.688465 11.888465
> x[x<fn[2]-c*(fn[4]-fn[2]) | x>fn[4]+c*(fn[4]-fn[2])]
[1] 71.6
> boxplot.stats(x,coef=c)$out
[1] 71.6

> x
[1] 1.0 2.3 5.0 6.7 8.0
> c<-2.6
> fn<-fivenum(x)
> fn
[1] 1.0 2.3 5.0 6.7 8.0
> boxplot.stats(x,coef=c)$stats
[1] 1.0 2.3 5.0 6.7 8.0
> n<-length(x)
> n
[1] 5
> boxplot.stats(x,coef=c)$n
[1] 5
> median(x)+c(-1,1)*1.58*(fn[4]-fn[2])/sqrt(n)
[1] 1.890971 8.109029
> boxplot.stats(x,coef=c)$conf
[1] 1.890971 8.109029
> x[x<fn[2]-c*(fn[4]-fn[2]) | x>fn[4]+c*(fn[4]-fn[2])]
numeric(0)
> boxplot.stats(x,coef=c)$out
numeric(0)
```

- **Osservazioni:** Calcola i quartili con la funzione `fivenum()`.

3.8 Funzioni di distribuzione di frequenza

tabulate()

- **Package:** `base`
- **Parametri:**

  `bin` vettore di valori naturali di dimensione n

- **Significato:** distribuzione di frequenza per i valori naturali 1, 2, ..., max(`bin`)
- **Esempio:**

```
> tabulate(bin=c(2,3,5))
[1] 0 1 1 0 1

> tabulate(bin=c(2,3,3,5))
[1] 0 1 2 0 1

> tabulate(bin=c(-2,0,2,3,3,5))
[1] 0 1 2 0 1
```

table()

- **Package:** `base`
- **Parametri:**

  `x` vettore alfanumerico di dimensione n

- **Significato:** distribuzione di frequenza
- **Esempio:**

```
> x
[1] "a" "a" "b" "c" "a" "c"
> table(x)  # frequenza assoluta
x
a b c
3 1 2
> table(x)/length(x)  # frequenza relativa
x
        a         b         c
0.5000000 0.1666667 0.3333333

> f
[1] a b c b a c a b b c a
Levels: a b c
> g
[1] A S A S S S A S S A A
Levels: A S
> table(f,g)
   g
f   A S
  a 3 1
  b 0 4
  c 2 1

> x
[1] 1 2 3 2 1 3 1 1 2 3
> table(x)
x
1 2 3
4 3 3
```

unique()

- **Package:** `base`
- **Parametri:**

  `x` vettore alfanumerico di dimensione n

- **Significato:** supporto (valori distinti di x)
- **Esempio:**

```
> x
[1] "a" "a" "b" "c" "a" "c"
> unique(x)
a b c

> x
[1] 1 2 3 2 1 3 1 1 2 3
> unique(x)
[1] 1 2 3
```

n.bins()

- **Package:** `car`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `rule = freedman.diaconis / sturges / scott / simple` algoritmo

- **Significato:** algoritmo di calcolo per il numero di classi di un istogramma
- **Formula:**

`rule = freedman.diaconis`

$$\left\lceil \frac{x_{(n)} - x_{(1)}}{2\,(Q_{0.75}(x) - Q_{0.25}(x))\,n^{-1\,/\,3}} \right\rceil$$

`rule = sturges`

$$\lceil \log_2(n) + 1 \rceil$$

`rule = scott`

$$\left\lceil \frac{x_{(n)} - x_{(1)}}{3.5\,s_x\,n^{-1\,/\,3}} \right\rceil$$

`rule = simple`

$$\begin{cases} \lfloor 2\sqrt{n} \rfloor & \text{se } n \leq 100 \\ \lfloor 10\,\log_{10}(n) \rfloor & \text{se } n > 100 \end{cases}$$

- **Esempio:**

```
> x
[1] 1.0 2.3 5.0 6.7 8.0
> x<-sort(x)
> n<-length(x)
> n
[1] 5
> Q1<-quantile(x,probs=0.25)
```

```
> Q3<-quantile(x,probs=0.75)
> as.numeric(ceiling((x[n]-x[1])/(2*(Q3-Q1)*n^(-1/3))))
[1] 2
> n.bins(x,rule="freedman.diaconis")
[1] 2

> x
[1] 1.0 2.3 5.0 6.7 8.0
> n<-length(x)
> n
[1] 5
> ceiling(log2(n)+1)
[1] 4
> n.bins(x,rule="sturges")
[1] 4

> x
[1] 1.0 2.3 5.0 6.7 8.0
> x<-sort(x)
> n<-length(x)
> n
[1] 5
> sx<-sd(x)
> ceiling((x[n]-x[1])/(3.5*sx*n^(-1/3)))
[1] 2
> n.bins(x,rule="scott")
[1] 2

> x
[1] 1.0 2.3 5.0 6.7 8.0
> n<-length(x)
> n
[1] 5
> # n <= 100
> floor(2*sqrt(n))
[1] 4
> n.bins(x,rule="simple")
[1] 4
```

- **Osservazioni:** Calcola i quartili con la funzione `quantile()`.

nclass.FD()

- **Package:** `grDevices`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** numero di classi di un istogramma secondo *Freedman - Diaconis*
- **Formula:**

$$\left\lceil \frac{x_{(n)} - x_{(1)}}{2\,(Q_{0.75}(x) - Q_{0.25}(x))\,n^{-1\,/\,3}} \right\rceil$$

- **Esempio:**

```
> x
[1] 1.0 2.3 5.0 6.7 8.0
> x<-sort(x)
> n<-length(x)
> n
[1] 5
```

```
> Q1<-quantile(x,probs=0.25)
> Q3<-quantile(x,probs=0.75)
> as.numeric(ceiling((x[n]-x[1])/(2*(Q3-Q1)*n^(-1/3))))
[1] 2
> nclass.FD(x)
[1] 2
```

- **Osservazioni:** Calcola i quartili con la funzione `quantile()`.

nclass.Sturges()

- **Package:** `grDevices`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** numero di classi di un istogramma secondo *Sturges*
- **Formula:**

$$\lceil \log_2(n) + 1 \rceil$$

- **Esempio:**

```
> x
[1] 1.0 2.3 5.0 6.7 8.0
> n<-length(x)
> n
[1] 5
> ceiling(log2(n)+1)
[1] 4
> nclass.Sturges(x)
[1] 4
```

nclass.scott()

- **Package:** `grDevices`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** numero di classi di un istogramma secondo *Scott*
- **Formula:**

$$\left\lceil \frac{x_{(n)} - x_{(1)}}{3.5\, s_x\, n^{-1/3}} \right\rceil$$

- **Esempio:**

```
> x
[1] 1.0 2.3 5.0 6.7 8.0
> x<-sort(x)
> n<-length(x)
> n
[1] 5
> sx<-sd(x)
> ceiling((x[n]-x[1])/(3.5*sx*n^(-1/3)))
[1] 2
> nclass.scott(x)
[1] 2
```

hist()

- **Package:** `graphics`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `breaks` estremi delle classi di ampiezza b_i

  `right = T / F` classi chiuse a destra $\left(a_{(i)},\, a_{(i+1)}\right]$ oppure a sinistra $\left[a_{(i)},\, a_{(i+1)}\right)$

  `include.lowest = T / F` estremo incluso

- **Significato:** istogramma
- **Output:**

  `breaks` estremi delle classi

  `counts` frequenze assolute

  `density` densità di frequenza

  `mids` punti centrali delle classi

- **Formula:**

  `breaks`

  $$a_{(i)} \quad \forall\, i = 1, 2, \ldots, m$$

  `counts`

  $$n_i \quad \forall\, i = 1, 2, \ldots, m-1$$

  `density`

  $$\frac{n_i}{n\, b_i} \quad \forall\, i = 1, 2, \ldots, m-1$$

  `mids`

  $$\frac{a_{(i)} + a_{(i+1)}}{2} \quad \forall\, i = 1, 2, \ldots, m-1$$

- **Esempio:**

```
> x
[1] 51.10 52.30 66.70 77.10 77.15 77.17
> n<-length(x)
> n
[1] 6
> m<-4
> a1<-50
> a2<-65
> a3<-70
> a4<-85
> a<-c(a1,a2,a3,a4)
> b1<-65-50
> b2<-70-65
> b3<-85-70
> b<-c(b1,b2,b3)
> b
[1] 15  5 15
> hist(x,breaks=a,right=F,include.lowest=F,plot=F)$breaks
[1] 50 65 70 85
> count<-numeric(m-1)
> count[1]<-sum(x>=a1 & x<a2)
> count[2]<-sum(x>=a2 & x<a3)
> count[3]<-sum(x>=a3 & x<a4)
> count
[1] 2 1 3
> hist(x,breaks=a,right=F,include.lowest=F,plot=F)$counts
[1] 2 1 3
> count/(n*b)
```

```
[1] 0.02222222 0.03333333 0.03333333
> hist(x,breaks=a,right=F,include.lowest=F,plot=F)$density
[1] 0.02222222 0.03333333 0.03333333
> (a[-m]+a[-1])/2
[1] 57.5 67.5 77.5
> hist(x,breaks=a,right=F,include.lowest=F,plot=F)$mids
[1] 57.5 67.5 77.5

> x
[1]  1.0  1.2  2.2  2.3  3.0  5.0  6.7  8.0 15.6
> n<-length(x)
> n
[1] 9
> m<-5
> a1<-0
> a2<-5
> a3<-10
> a4<-15
> a5<-20
> a<-c(a1,a2,a3,a4,a5)
> a
[1]  0  5 10 15 20
> b1<-a2-a1
> b2<-a3-a2
> b3<-a4-a3
> b4<-a5-a4
> b<-c(b1,b2,b3,b4)
> b
[1] 5 5 5 5
> hist(x,breaks=a,right=F,include.lowest=F,plot=F)$breaks
[1]  0  5 10 15 20
> count<-numeric(m-1)
> count[1]<-sum(x>=a1 & x<a2)
> count[2]<-sum(x>=a2 & x<a3)
> count[3]<-sum(x>=a3 & x<a4)
> count[4]<-sum(x>=a4 & x<a5)
> count
[1] 5 3 0 1
> hist(x,breaks=a,right=F,include.lowest=F,plot=F)$counts
[1] 5 3 0 1
> count/(n*b)
[1] 0.11111111 0.06666667 0.00000000 0.02222222
> hist(x,breaks=a,right=F,include.lowest=F,plot=F)$density
[1] 0.11111111 0.06666667 0.00000000 0.02222222
> (a[-m]+a[-1])/2
[1]  2.5  7.5 12.5 17.5
> hist(x,breaks=a,right=F,include.lowest=F,plot=F)$mids
[1]  2.5  7.5 12.5 17.5
```

cut()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `breaks` estremi delle classi di ampiezza b_i

  `right = T / F` classi chiuse a destra $\left(a_{(i)},\, a_{(i+1)}\right]$ oppure a sinistra $\left[a_{(i)},\, a_{(i+1)}\right)$

  `include.lowest = T / F` estremo incluso

  `labels` etichette

- **Significato:** raggruppamento in classi
- **Esempio:**

```
> x
[1] 1.20 2.30 4.50 5.40 3.40 5.40 2.30 2.10 1.23 4.30 0.30
> n<-length(x)
> n
[1] 11
> cut(x,breaks=c(0,4,6),right=T,include.lowest=F,labels=c("0-4","4-6"))
[1] 0-4 0-4 4-6 4-6 0-4 4-6 0-4 0-4 0-4 4-6 0-4
Levels: 0-4 4-6

> x
[1] 1.0 2.0 3.0 4.0 5.6 7.4 1.2 4.0 4.4
> n<-length(x)
> n
[1] 9
> cut(x,breaks=c(0,4,8),right=T,include.lowest=F,labels=c("0-4","4-8"))
[1] 0-4 0-4 0-4 0-4 4-8 4-8 0-4 0-4 4-8
Levels: 0-4 4-8
```

3.9 Funzioni di adattamento normale

qqnorm()

- **Package:** `stats`
- **Parametri:**

  `y` vettore numerico di dimensione n ordinato in maniera crescente

- **Significato:** quantili teorici e campionari per QQ-Norm
- **Output:**

  `x` quantili teorici

  `y` quantili campionari

- **Formula:**

  `x`

$$\begin{cases} \Phi^{-1}\left((8\,i-3)\,/\,(8\,n+2)\right) & \forall\, i\,=\,1,\,2,\,\ldots,\,n \qquad \text{se } n \leq 10 \\ \Phi^{-1}\left((i-1\,/\,2)\,/\,n\right) & \forall\, i\,=\,1,\,2,\,\ldots,\,n \qquad \text{se } n > 10 \end{cases}$$

  `y`

$$y_{(i)} \quad \forall\, i\,=\,1,\,2,\,\ldots,\,n$$

- **Esempio:**

```
> y
[1]  3.2  1.4  4.2 12.4 13.4 17.3 18.1
> y<-sort(y)
> y
[1]  1.4 3.2  4.2 12.4 13.4 17.3 18.1
> n<-length(y)
> n
[1] 7
> # 7 <= 10
> qqnorm(y)$y
[1]  1.4  3.2  4.2 12.4 13.4 17.3 18.1
> qnorm((8*(1:n)-3)/(8*n+2))
[1] -1.36448875 -0.75829256 -0.35293399  0.00000000  0.35293399
```

```
[6]  0.75829256  1.36448875
> qqnorm(y)$x
[1] -1.36448875 -0.75829256 -0.35293399  0.00000000  0.35293399
[6]  0.75829256  1.36448875

> y
[1]  1.20  2.30  4.30 -3.40  4.20  5.43  3.20  2.20  0.20  2.10
[11] 2.20  3.10
> y<-sort(y)
> y
[1] -3.40  0.20  1.20  2.10  2.20  2.20  2.30  3.10  3.20  4.20
[11] 4.30  5.43
> n<-length(y)
> n
[1] 12
> # 12 > 10
> qqnorm(y)$y
[1] -3.40  0.20  1.20  2.10  2.20  2.20  2.30  3.10  3.20  4.20
[11] 4.30  5.43
> qnorm(((1:n)-1/2)/n)
[1] -1.73166440 -1.15034938 -0.81221780 -0.54852228 -0.31863936
[6] -0.10463346  0.10463346  0.31863936  0.54852228  0.81221780
[11]  1.15034938  1.73166440
> qqnorm(y)$x
 [1] -1.73166440 -1.15034938 -0.81221780 -0.54852228 -0.31863936
 [6] -0.10463346  0.10463346  0.31863936  0.54852228  0.81221780
[11]  1.15034938  1.73166440
```

ppoints()

- **Package:** `stats`
- **Parametri:**

  `n` valore naturale

- **Significato:** rapporti per QQ-Norm
- **Formula:**

$$\begin{cases} (8\,i-3)\,/\,(8\,n+2) & \forall\, i\,=\,1,\,2,\,\ldots,\,n \qquad \text{se } n \leq 10 \\ (i-1\,/\,2)\,/\,n & \forall\, i\,=\,1,\,2,\,\ldots,\,n \qquad \text{se } n > 10 \end{cases}$$

- **Esempio:**

```
> n
[1] 5
> # 5 <= 10
> (8*(1:n)-3)/(8*n+2)
[1] 0.11904762 0.30952381 0.50000000 0.69047619 0.88095238
> ppoints(n=5)
[1] 0.11904762 0.30952381 0.50000000 0.69047619 0.88095238

> n
[1] 12
> # 12 > 10
> ((1:n)-1/2)/n
[1] 0.041666667 0.125000000 0.208333333 0.291666667 0.375000000
[6] 0.458333333 0.541666667 0.625000000 0.708333333 0.791666667
[11] 0.875000000 0.958333333
> ppoints(n=12)
[1] 0.041666667 0.125000000 0.208333333 0.291666667 0.375000000
[6] 0.458333333 0.541666667 0.625000000 0.708333333 0.791666667
[11] 0.875000000 0.958333333
```

3.10 Funzioni logistiche

logit()

- **Package:** `faraway`
- **Parametri:**

  `x` vettore numerico di probabilità di dimensione n

- **Significato:** trasformazione logit
- **Formula:**

$$\log\left(\frac{x_i}{1-x_i}\right) \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> x
[1] 0.20 0.34 0.54 0.65 0.11
> log(x/(1-x))
[1] -1.3862944 -0.6632942  0.1603427  0.6190392 -2.0907411
> logit(x)
[1] -1.3862944 -0.6632942  0.1603427  0.6190392 -2.0907411

> x
[1] 0.23 0.45 0.67 0.89 0.11
> log(x/(1-x))
[1] -1.2083112 -0.2006707  0.7081851  2.0907411 -2.0907411
> logit(x)
[1] -1.2083112 -0.2006707  0.7081851  2.0907411 -2.0907411
```

ilogit()

- **Package:** `faraway`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** trasformazione logit inversa
- **Formula:**

$$\frac{e^{x_i}}{1+e^{x_i}} \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> x
[1]  1  2  3  5 -6
> exp(x)/(1+exp(x))
[1] 0.731058579 0.880797078 0.952574127 0.993307149 0.002472623
> ilogit(x)
[1] 0.731058579 0.880797078 0.952574127 0.993307149 0.002472623

> x
[1]  2.3  4.5  6.7  7.8 12.0
> exp(x)/(1+exp(x))
[1] 0.9088770 0.9890131 0.9987706 0.9995904 0.9999939
> ilogit(x)
[1] 0.9088770 0.9890131 0.9987706 0.9995904 0.9999939
```

inv.logit()

- **Package:** `boot`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** trasformazione logit inversa
- **Formula:**

$$\frac{e^{x_i}}{1+e^{x_i}} \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> x
[1]  1  2  3  5 -6
> exp(x)/(1+exp(x))
[1] 0.731058579 0.880797078 0.952574127 0.993307149 0.002472623
> inv.logit(x)
[1] 0.731058579 0.880797078 0.952574127 0.993307149 0.002472623

> x
[1]  2.3  4.5  6.7  7.8 12.0
> exp(x)/(1+exp(x))
[1] 0.9088770 0.9890131 0.9987706 0.9995904 0.9999939
> ilogit(x)
[1] 0.9088770 0.9890131 0.9987706 0.9995904 0.9999939
```

3.11 Funzioni di distribuzione discrete

Bernoulli

$p_X(x) = p^x (1-p)^{1-x} \quad x = 0, 1, \quad 0 < p < 1$

$\mu_X = p$

$\sigma_X^2 = p(1-p)$

Binomiale

$p_X(x) = \binom{m}{x} p^x (1-p)^{m-x} \quad x = 0, 1, 2, \ldots, m, \quad m \in \mathbb{N} / \{0\}, \quad 0 < p < 1$

$\mu_X = m\,p$

$\sigma_X^2 = m\,p\,(1-p)$

Geometrica

$p_X(x) = p(1-p)^x \quad x \in \mathbb{N}, \quad 0 < p < 1$

$\mu_X = (1-p)/p$

$\sigma_X^2 = (1-p)/p^2$

Geometrica 2

$p_X(x) = p(1-p)^{x-1} \quad x \in \mathbb{N}\backslash\{0\}, \quad 0 < p < 1$

$\mu_X = 1/p$

$\sigma_X^2 = (1-p)/p^2$

Poisson

$p_X(x) = \lambda^x \, e^{-\lambda} \, / \, x! \quad x \in \mathbb{N}, \quad \lambda > 0$

$\mu_X = \lambda$

$\sigma_X^2 = \lambda$

Binomiale Negativa

$p_X(x) = \binom{r+x-1}{x} \, p^r \, (1-p)^x \quad x \in \mathbb{N}, \quad r \in \mathbb{N} \backslash \{0\}, \quad 0 < p < 1$

$\mu_X = r \, (1-p) \, / \, p$

$\sigma_X^2 = r \, (1-p) \, / \, p^2$

Ipergeometrica

$p_X(x) = \binom{M}{x} \binom{N-M}{k-x} / \binom{N}{k}$

$x = 0, 1, 2, \ldots, k$

$N \in \mathbb{N} \backslash \{0\}$

$k = 1, 2, \ldots, N$

$M = 0, 1, 2, \ldots, N-1$

$\mu_X = k \, (M \, / \, N)$

$\sigma_X^2 = k \, (M \, / \, N) \, (1 - M \, / \, N) \, (N-k) \, / \, (N-1)$

Multinomiale

$p_{X_1, X_2, \ldots, X_k}(x_1, x_2, \ldots, x_k) = \frac{m \, !}{x_1 \, ! \, x_2 \, ! \cdots x_k \, !} \prod_{i=1}^{k} p_i^{x_i}$

$x_i = 0, 1, 2, \ldots, m \quad \forall i = 1, 2, \ldots, k$

$0 < p_i < 1 \quad \forall i = 1, 2, \ldots, k$

$\sum_{i=1}^{k} x_i = m$

$\sum_{i=1}^{k} p_i = 1$

$\mu_{X_i} = m \, p_i \quad \forall i = 1, 2, \ldots, k$

$\sigma_{X_i}^2 = m \, p_i \, (1 - p_i) \quad \forall i = 1, 2, \ldots, k$

$\sigma_{X_i X_j} = -m \, p_i \, p_j \quad \forall i \neq j = 1, 2, \ldots, k$

Tavola argomenti comandi R

Variabile Casuale	Nome	Parametri	Package
Bernoulli	`binom`	`size, prob`	`stats`
Binomiale	`binom`	`size, prob`	`stats`
Geometrica	`geom`	`prob`	`stats`
Geometrica 2	`geomet`	`p`	`distributions`
Poisson	`pois`	`lambda`	`stats`
Binomiale Negativa	`nbinom`	`size, prob`	`stats`
Ipergeometrica	`hyper`	`m, n, k`	`stats`
Multinomiale	`multinom`	`size, prob`	`stats`

Tavola esempi comandi R

Variabile Casuale	Oggetto	Comando in R
Bernoulli	**Densità**	`dbinom(x=`x`,size=1,prob=`p`)`
	Ripartizione	`pbinom(q=`x`,size=1,prob=`p`)`
	Quantile	`qbinom(p=`α`,size=1,prob=`p`)`
	Estrazione random	`rbinom(n,size=1,prob=`p`)`
Binomiale	**Densità**	`dbinom(x=`x`,size=`m`,prob=`p`)`
	Ripartizione	`pbinom(q=`x`,size=`m`,prob=`p`)`
	Quantile	`qbinom(p=`α`,size=`m`,prob=`p`)`
	Estrazione random	`rbinom(n,size=`m`,prob=`p`)`
Geometrica	**Densità**	`dgeom(x=`x`,prob=`p`)`
	Ripartizione	`pgeom(q=`x`,prob=`p`)`
	Quantile	`qgeom(p=`α`,prob=`p`)`
	Estrazione random	`rgeom(n,prob=`p`)`
Geometrica 2	**Densità**	`geometpdf(p=`p`,x=`x`)`
	Ripartizione	`geometcdf(p=`p`,x=`x`)`
Poisson	**Densità**	`dpois(x=`x`,lambda=`λ`)`
	Ripartizione	`ppois(q=`x`,lambda=`λ`)`
	Quantile	`qpois(p=`α`,lambda=`λ`)`
	Estrazione random	`rpois(n,lambda=`λ`)`
Binomiale Negativa	**Densità**	`dnbinom(x=`x`,size=`r`,prob=`p`)`
	Ripartizione	`pnbinom(q=`x`,size=`r`,prob=`p`)`
	Quantile	`qnbinom(p=`α`,size=`r`,prob=`p`)`
	Estrazione random	`rnbinom(n,size=`r`,prob=`p`)`
Ipergeometrica	**Densità**	`dhyper(x=`x`,m=`M`,n=`$N-M$`,k=`k`)`
	Ripartizione	`phyper(q=`x`,m=`M`,n=`$N-M$`,k=`k`)`
	Quantile	`qhyper(p=`α`,m=`M`,n=`$N-M$`,k=`k`)`
	Estrazione random	`rhyper(nn,m=`M`,n=`$N-M$`,k=`k`)`
Multinomiale	**Densità**	`dmultinom(x=c(`$x_1,\ldots,x_k$`),prob=c(`$p_1,\ldots,p_k$`))`
	Estrazione random	`rmultinom(n,size=`m`,prob=c(`$p_1,\ldots,p_k$`))`

3.12 Funzioni di distribuzione continue

Normale

$f_X(x) = \left(2\,\pi\,\sigma^2\right)^{-1\,/\,2} \exp\left(-(x-\mu)^2\,/\,(2\,\sigma^2)\right) \quad x \in \mathbb{R}, \quad \mu \in \mathbb{R}, \quad \sigma > 0$

$\mu_X = \mu$

$\sigma_X^2 = \sigma^2$

Student

$f_X(x) = \frac{\Gamma((k+1)\,/\,2)}{\Gamma(k\,/\,2)}\,(k\,\pi)^{-1\,/\,2}\,(1+x^2\,/\,k)^{-(k+1)\,/\,2} \quad x \in \mathbb{R}, \quad k > 0$

$\mu_X = 0 \quad \text{per } k > 1$

$\sigma_X^2 = k\,/\,(k-2) \quad \text{per } k > 2$

Student non centrale

$f_X(x) = \frac{k^{k/2}\exp(-\delta^2/2)}{\sqrt{\pi}\,\Gamma(n/2)\,(k+x^2)^{(k+1)/2}} \sum_{i=0}^{\infty} \frac{\Gamma((k+i+1)/2)\,\delta^i}{i!} \left(\frac{2x^2}{k+x^2}\right)^{i/2} \quad x \in \mathbb{R}, \quad k > 0, \quad \delta \in \mathbb{R}$

$\mu_X = \sqrt{k/2}\,\delta\,\Gamma((k-1)/2)\,/\,\Gamma(k/2) \quad \text{per } k > 1$

$\sigma_X^2 = k(1+\delta^2)/(k-2) - \delta(k/2)\left(\Gamma((k-1)/2)\,/\,\Gamma(k/2)\right)^2 \quad \text{per } k > 2$

Chi - Quadrato

$f_X(x) = \frac{2^{-k/2}}{\Gamma(k/2)}\,x^{(k-2)/2}\,e^{-x/2} \quad x > 0, \quad k > 0$

$\mu_X = k$

$\sigma_X^2 = 2k$

Chi - Quadrato non centrale

$f_X(x) = \exp(-(x+\delta)/2) \sum_{i=0}^{\infty} \frac{(\delta/2)^i\,x^{k/2+i-1}}{2^{k/2+i}\,\Gamma(k/2+i)\,i!} \quad x > 0, \quad k > 0, \quad \delta > 0$

$\mu_X = k + \delta$

$\sigma_X^2 = 2(k+2\delta)$

Fisher

$f_X(x) = \frac{\Gamma((n_1+n_2)/2)}{\Gamma(n_1/2)\,\Gamma(n_2/2)} \left(\frac{n_1}{n_2}\right)^{n_1/2} x^{(n_1-2)/2} \left(1+\frac{n_1}{n_2}\,x\right)^{-(n_1+n_2)/2} \quad x,\, n_1,\, n_2 > 0$

$\mu_X = \frac{n_2}{n_2-2} \quad \text{per } n_2 > 2$

$\sigma_X^2 = \frac{2\,n_2^2\,(n_1+n_2-2)}{n_1\,(n_2-2)^2\,(n_2-4)} \quad \text{per } n_2 > 4$

Fisher non centrale

$f_X(x) = \frac{n_1^{n_1/2}\,n_2^{n_2/2}}{\exp(\delta/2)} \frac{x^{n_1/2-1}}{(n_1\,x+n_2)^{(n_1+n_2)/2}} \sum_{i=0}^{\infty} \frac{(\delta/2)^i}{i!} \frac{\Gamma(n_1/2+n_2/2+i)}{\Gamma(n_1/2+i)\,\Gamma(n_2/2)} \left(\frac{n_1\,x}{n_1\,x+n_2}\right)^i \quad x,\, n_1,\, n_2,\, \delta > 0$

$\mu_X = \frac{n_2\,(n_1+\delta)}{n_1\,(n_2-2)} \quad \text{per } n_2 > 2$

$\sigma_X^2 = 2\left(\frac{n_2}{n_1}\right)^2 \frac{(n_1+\delta)^2+(n_1+2\,\delta)\,(n_2-2)}{(n_2-2)^2\,(n_2-4)} \quad \text{per } n_2 > 4$

Esponenziale

$f_X(x) = \lambda\,e^{-\lambda\,x} \quad x > 0, \quad \lambda > 0$

$\mu_X = 1/\lambda$

$\sigma_X^2 = 1/\lambda^2$

Gamma

$f_X(x) = \frac{\lambda^\theta}{\Gamma(\theta)}\,x^{\theta-1}\,e^{-\lambda\,x} \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X = \theta/\lambda$

$\sigma_X^2 = \theta/\lambda^2$

Gamma 2

$f_X(x) = \frac{1}{\lambda^\theta\,\Gamma(\theta)}\,x^{\theta-1}\,e^{-x/\lambda} \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X = \theta\,\lambda$

$\sigma_X^2 = \theta\,\lambda^2$

Gamma inversa

$f_X(x) = \frac{\lambda^\theta}{\Gamma(\theta)} x^{-(\theta+1)} e^{-\lambda/x} \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X = \lambda/(\theta-1) \quad \text{per } \theta > 1$

$\sigma_X^2 = \lambda^2/[(\theta-1)^2(\theta-2)] \quad \text{per } \theta > 2$

Gamma inversa 2

$f_X(x) = \frac{1}{\lambda^\theta \Gamma(\theta)} x^{-(\theta+1)} e^{-1/(\lambda x)} \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X = 1/[\lambda(\theta-1)] \quad \text{per } \theta > 1$

$\sigma_X^2 = 1/[\lambda^2(\theta-1)^2(\theta-2)] \quad \text{per } \theta > 2$

LogNormale

$f_X(x) = \left(\sigma x \sqrt{2\pi}\right)^{-1} \exp\left(-(\log(x)-\mu)^2/(2\sigma^2)\right) \quad x > 0, \quad \mu \in \mathbb{R}, \sigma > 0$

$\mu_X = \exp(\mu + \sigma^2/2)$

$\sigma_X^2 = \exp(2\mu+\sigma^2)\left(\exp\left(\sigma^2\right)-1\right)$

Weibull

$f_X(x) = (\theta/\lambda)(x/\lambda)^{\theta-1} \exp\left(-(x/\lambda)^\theta\right) \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X = \lambda\Gamma((\theta+1)/\theta)$

$\sigma_X^2 = \lambda^2\left[\Gamma((\theta+2)/\theta) - \Gamma^2((\theta+1)/\theta)\right]$

Beta

$f_X(x) = \frac{\Gamma(\theta+\lambda)}{\Gamma(\theta)\Gamma(\lambda)} x^{\theta-1}(1-x)^{\lambda-1} \quad 0 < x < 1, \quad \theta > 0, \quad \lambda > 0$

$\mu_X = \theta/(\theta+\lambda)$

$\sigma_X^2 = \theta\lambda/\left[(\theta+\lambda+1)(\theta+\lambda)^2\right]$

Beta non centrale

$\frac{\chi^2_\theta(\delta)}{\chi^2_\theta(\delta)+\chi^2_\lambda} \quad 0 < x < 1, \quad \theta > 0, \quad \lambda > 0, \quad \delta > 0$

Logistica

$f_X(x) = \lambda^{-1}\exp((x-\theta)/\lambda)\left(1+\exp((x-\theta)/\lambda)\right)^{-2} \quad x \in \mathbb{R}, \quad \theta \in \mathbb{R}, \quad \lambda > 0$

$\mu_X = \theta$

$\sigma_X^2 = (\pi\lambda)^2/3$

Cauchy

$f_X(x) = (\pi\lambda)^{-1}\left[1+((x-\theta)/\lambda)^2\right]^{-1} \quad x \in \mathbb{R}, \quad \theta \in \mathbb{R}, \quad \lambda > 0$

$\mu_X = \nexists$

$\sigma_X^2 = \nexists$

Uniforme

$f_X(x) = 1/(b-a) \quad a < x < b, \quad a \in \mathbb{R}, \quad b \in \mathbb{R}, \quad a < b$

$\mu_X = (a+b)/2$

$\sigma_X^2 = (b-a)^2/12$

Normale inversa - Wald

$f_X(x) = (\lambda / (2\pi x^3))^{1/2} \exp\left(-\lambda (x-\theta)^2 / (2\theta^2 x)\right) \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X = \theta$

$\sigma_X^2 = \theta^3 / \lambda$

Wilcoxon signed rank

$0 \leq x \leq n(n+1)/2, \quad n \in \mathbb{N} / \{0\}$

$\mu_X = n(n+1)/4$

$\sigma_X^2 = n(n+1)(2n+1)/24$

Mann - Whitney

$0 \leq x \leq n_x n_y, \quad n_x \in \mathbb{N} / \{0\}, \quad n_y \in \mathbb{N} / \{0\}$

$\mu_X = n_x n_y / 2$

$\sigma_X^2 = n_x n_y (n_x + n_y + 1)/12$

Dirichlet

$f_{X_1,X_2,\ldots,X_k}(x_1, x_2, \ldots, x_k) = \frac{\Gamma(\alpha_1+\alpha_2+\cdots+\alpha_k)}{\Gamma(\alpha_1)\Gamma(\alpha_2)\cdots\Gamma(\alpha_k)} \prod_{i=1}^{k} x_i^{\alpha_i - 1}$

$x_i \geq 0 \quad \forall i = 1, 2, \ldots, k$

$\alpha_i > 0 \quad \forall i = 1, 2, \ldots, k$

$\sum_{i=1}^{k} x_i = 1$

$\sum_{i=1}^{k} \alpha_i = \alpha$

$\mu_{X_i} = \frac{\alpha_i}{\alpha} \quad \forall i = 1, 2, \ldots, k$

$\sigma_{X_i}^2 = \frac{\alpha_i(\alpha-\alpha_i)}{\alpha^2(\alpha+1)} \quad \forall i = 1, 2, \ldots, k$

$\sigma_{X_i X_j} = -\frac{\alpha_i \alpha_j}{\alpha^2(\alpha+1)} \quad \forall i \neq j = 1, 2, \ldots, k$

Normale doppia

$f_{X_1,X_2}(x_1, x_2) = \frac{1}{2\pi \sigma_{11}^{\frac{1}{2}} \sigma_{22}^{\frac{1}{2}} \sqrt{1-\rho_{12}^2}} \exp\left(-\frac{1}{2(1-\rho_{12}^2)}\left[\left(\frac{x_1-\mu_1}{\sigma_{11}^{\frac{1}{2}}}\right)^2 - 2\rho_{12}\frac{x_1-\mu_1}{\sigma_{11}^{\frac{1}{2}}}\frac{x_2-\mu_2}{\sigma_{22}^{\frac{1}{2}}} + \left(\frac{x_2-\mu_2}{\sigma_{22}^{\frac{1}{2}}}\right)^2\right]\right)$

$x_i \in \mathbb{R} \quad \forall i = 1, 2$

$\mu_i \in \mathbb{R} \quad \forall i = 1, 2$

$\rho_{12} \in (0, 1)$

$V = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix}$ definita positiva

$\sigma_{ii} > 0 \quad \forall i = 1, 2$

$\mu_{X_i} = \mu_i \quad \forall i = 1, 2$

$\sigma_{X_i}^2 = \sigma_{ii} \quad \forall i = 1, 2$

$\sigma_{X_1 X_2} = \sigma_{12}$

Normale multipla

$f_{X_1,\ldots,X_k}(x_1, x_2, \ldots, x_k) = \frac{1}{(2\pi)^{k/2}\sqrt{\det(V)}} \exp\left(-\frac{1}{2}(x_1-\mu_1, x_2-\mu_2, \ldots, x_k-\mu_k)^T V^{-1} (x_1-\mu_1, x_2-\mu_2, \ldots, x_k-\mu_k)\right)$

$x_i \in \mathbb{R} \quad \forall i = 1, 2, \ldots, k$

$\mu_i \in \mathbb{R} \quad \forall i = 1, 2, \ldots, k$

$$V = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \ldots & \sigma_{1k} \\ \sigma_{21} & \sigma_{22} & \ldots & \sigma_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{k1} & \sigma_{k2} & \ldots & \sigma_{kk} \end{pmatrix} \quad \text{definita positiva}$$

$\sigma_{ii} > 0 \quad \forall i = 1, 2, \ldots, k$

$\mu_{X_i} = \mu_i \quad \forall i = 1, 2, \ldots, k$

$\sigma^2_{X_i} = \sigma^2_{ii} \quad \forall i = 1, 2, \ldots, k$

$\sigma_{X_i X_j} = \sigma_{ij} \quad \forall i \neq j = 1, 2, \ldots, k$

Tavola argomenti comandi R

Variabile Casuale	Nome	Parametri	Package
Normale	`norm`	`mean, sd`	`stats`
Student	`t`	`df`	`stats`
Student non centrale	`t`	`df, ncp`	`stats`
Chi - Quadrato	`chisq`	`df`	`stats`
Chi - Quadrato non centrale	`chisq`	`df, ncp`	`stats`
Fisher	`f`	`df1, df2`	`stats`
Fisher non centrale	`f`	`df1, df2, ncp`	`stats`
Esponenziale	`exp`	`rate`	`stats`
Gamma	`gamma`	`shape, scale, rate`	`stats`
Gamma 2	`gamma`	`shape, scale, rate`	`stats`
Gamma inversa	`invgamma`	`shape, scale`	`MCMCpack`
Gamma inversa 2	`invgamma`	`shape, scale`	`MCMCpack`
LogNormale	`lnorm`	`meanlog, sdlog`	`stats`
Weibull	`weibull`	`shape, scale`	`stats`
Beta	`beta`	`shape1, shape2`	`stats`
Beta non centrale	`beta`	`shape1, shape2, ncp`	`stats`
Logistica	`logis`	`location, scale`	`stats`
Cauchy	`cauchy`	`location, scale`	`stats`
Uniforme	`unif`	`min, max`	`stats`
Normale inversa - Wald	`invGauss`	`nu, lambda`	`SuppDists`
Wilcoxon signed rank	`signrank`	`n`	`stats`
Mann - Whitney	`wilcox`	`m, n`	`stats`
Dirichlet	`dirichlet`	`alpha`	`MCMCpack`
Normale doppia	`mvnorm`	`mean, sigma`	`mvtnorm`
Normale multipla	`mvnorm`	`mean, sigma`	`mvtnorm`

Tavola esempi comandi R

Variabile Casuale	Oggetto	Comando in R
Normale	**Densità**	`dnorm(x=`x`,mean=`μ`,sd=`σ`)`
	Ripartizione	`pnorm(q=`x`,mean=`μ`,sd=`σ`)`
	Quantile	`qnorm(p=`α`,mean=`μ`,sd=`σ`)`
	Estrazione random	`rnorm(n,mean=`μ`,sd=`σ`)`
Student	**Densità**	`dt(x=`x`,df=`k`)`
	Ripartizione	`pt(q=`x`,df=`k`)`
	Quantile	`qt(p=`α`,df=`k`)`
	Estrazione random	`rt(n,df=`k`)`
Student non centrale	**Densità**	`dt(x=`x`,df=`k`,ncp=`δ`)`
	Ripartizione	`pt(q=`x`,df=`k`,ncp=`δ`)`
	Quantile	`qt(p=`α`,df=`k`,ncp=`δ`)`
	Estrazione random	`rt(n,df=`k`,ncp=`δ`)`

Chi - Quadrato	**Densità**	`dchisq(x=`x`,df=`k`)`
	Ripartizione	`pchisq(q=`x`,df=`k`)`
	Quantile	`qchisq(p=`α`,df=`k`)`
	Estrazione random	`rchisq(n,df=`k`)`
Chi - Quadrato non centrale	**Densità**	`dchisq(x=`x`,df=`k`,ncp=`δ`)`
	Ripartizione	`pchisq(q=`x`,df=`k`,ncp=`δ`)`
	Quantile	`qchisq(p=`α`,df=`k`,ncp=`δ`)`
	Estrazione random	`rchisq(n,df=`k`,ncp=`δ`)`
Fisher	**Densità**	`df(x=`x`,df1=`n_1`,df2=`n_2`)`
	Ripartizione	`pf(q=`x`,df1=`n_1`,df2=`n_2`)`
	Quantile	`qf(p=`α`,df1=`n_1`,df2=`n_2`)`
	Estrazione random	`rf(n,df1=`n_1`,df2=`n_2`)`
Fisher non centrale	**Densità**	`df(x=`x`,df1=`n_1`,df2=`n_2`,ncp=`δ`)`
	Ripartizione	`pf(q=`x`,df1=`n_1`,df2=`n_2`,ncp=`δ`)`
	Quantile	`qf(p=`α`,df1=`n_1`,df2=`n_2`,ncp=`δ`)`
	Estrazione random	`rf(n,df1=`n_1`,df2=`n_2`,ncp=`δ`)`
Esponenziale	**Densità**	`dexp(x=`x`,rate=`λ`)`
	Ripartizione	`pexp(q=`x`,rate=`λ`)`
	Quantile	`qexp(p=`α`,rate=`λ`)`
	Estrazione random	`rexp(n,rate=`λ`)`
Gamma	**Densità**	`dgamma(x=`x`,shape=`θ`,rate=`λ`)` `dgamma(x=`x`,shape=`θ`,scale=`$1/\lambda$`)`
	Ripartizione	`pgamma(q=`x`,shape=`θ`,rate=`λ`)` `pgamma(q=`x`,shape=`θ`,scale=`$1/\lambda$`)`
	Quantile	`qgamma(p=`α`,shape=`θ`,rate=`λ`)` `qgamma(p=`α`,shape=`θ`,scale=`$1/\lambda$`)`
	Estrazione random	`rgamma(n,shape=`θ`,rate=`λ`)` `rgamma(n,shape=`θ`,scale=`$1/\lambda$`)`
Gamma 2	**Densità**	`dgamma(x=`x`,shape=`θ`,rate=`$1/\lambda$`)` `dgamma(x=`x`,shape=`θ`,scale=`λ`)`
	Ripartizione	`pgamma(q=`x`,shape=`θ`,rate=`$1/\lambda$`)` `pgamma(q=`x`,shape=`θ`,scale=`λ`)`
	Quantile	`qgamma(p=`α`,shape=`θ`,rate=`$1/\lambda$`)` `qgamma(p=`α`,shape=`θ`,scale=`λ`)`
	Estrazione random	`rgamma(n,shape=`θ`,rate=`$1/\lambda$`)` `rgamma(n,shape=`θ`,scale=`λ`)`
Gamma inversa	**Densità**	`dinvgamma(x=`x`,shape=`θ`,scale=`$1/\lambda$`)`
	Estrazione random	`rinvgamma(n,shape=`θ`,scale=`λ`)`
Gamma inversa 2	**Densità**	`dinvgamma(x=`x`,shape=`θ`,scale=`λ`)`
	Estrazione random	`rinvgamma(n,shape=`θ`,scale=`$1/\lambda$`)`
LogNormale	**Densità**	`dlnorm(x=`x`,meanlog=`μ`,sdlog=`σ`)`
	Ripartizione	`plnorm(q=`x`,meanlog=`μ`,sdlog=`σ`)`
	Quantile	`qlnorm(p=`α`,meanlog=`μ`,sdlog=`σ`)`
	Estrazione random	`rlnorm(n,meanlog=`μ`,sdlog=`σ`)`
Weibull	**Densità**	`dweibull(x=`x`,shape=`θ`,scale=`λ`)`
	Ripartizione	`pweibull(q=`x`,shape=`θ`,scale=`λ`)`
	Quantile	`qweibull(p=`α`,shape=`θ`,scale=`λ`)`
	Estrazione random	`rweibull(n,shape=`θ`,scale=`λ`)`
Beta	**Densità**	`dbeta(x=`x`,shape1=`θ`,shape2=`λ`)`
	Ripartizione	`pbeta(q=`x`,shape1=`θ`,shape2=`λ`)`
	Quantile	`qbeta(p=`α`,shape1=`θ`,shape2=`λ`)`
	Estrazione random	`rbeta(n,shape1=`θ`,shape2=`λ`)`
Beta non centrale	**Densità**	`dbeta(x=`x`,shape1=`θ`,shape2=`λ`,ncp=`δ`)`
	Ripartizione	`pbeta(q=`x`,shape1=`θ`,shape2=`λ`,ncp=`δ`)`
	Quantile	`qbeta(p=`α`,shape1=`θ`,shape2=`λ`,ncp=`δ`)`
	Estrazione random	`rbeta(n,shape1=`θ`,shape2=`λ`,ncp=`δ`)`
Logistica	**Densità**	`dlogis(x=`x`,location=`θ`,scale=`λ`)`
	Ripartizione	`plogis(q=`x`,location=`θ`,scale=`λ`)`
	Quantile	`qlogis(p=`α`,location=`θ`,scale=`λ`)`
	Estrazione random	`rlogis(n,location=`θ`,scale=`λ`)`
Cauchy	**Densità**	`dcauchy(x=`x`,location=`θ`,scale=`λ`)`
	Ripartizione	`pcauchy(q=`x`,location=`θ`,scale=`λ`)`

	Quantile	`qcauchy(p=`α`,location=`θ`,scale=`λ`)`
	Estrazione random	`rcauchy(n,location=`θ`,scale=`λ`)`
Uniforme	**Densità**	`dunif(x=`x`,min=`a`,max=`b`)`
	Ripartizione	`punif(q=`x`,min=`a`,max=`b`)`
	Quantile	`qunif(p=`α`,min=`a`,max=`b`)`
	Estrazione random	`runif(n,min=`a`,max=`b`)`
Normale inversa - Wald	**Densità**	`dinvGauss(x=`x`,nu=`θ`,lambda=`λ`)`
	Ripartizione	`pinvGauss(q=`x`,nu=`θ`,lambda=`λ`)`
	Quantile	`qinvGauss(p=`α`,nu=`θ`,lambda=`λ`)`
	Estrazione random	`rinvGauss(n,nu=`θ`,lambda=`λ`)`
Wilcoxon signed rank	**Densità**	`dsignrank(x=`x`,n=`n`)`
	Ripartizione	`psignrank(q=`x`,n=`n`)`
	Quantile	`qsignrank(p=`α`,n=`n`)`
	Estrazione random	`rsignrank(nn,n=`n`)`
Mann - Whitney	**Densità**	`dwilcox(x=`x`,m=`n_x`,n=`n_y`)`
	Ripartizione	`pwilcox(q=`x`,m=`n_x`,n=`n_y`)`
	Quantile	`qwilcox(p=`α`,m=`n_x`,n=`n_y`)`
	Estrazione random	`rwilcox(nn,m=`n_x`,n=`n_y`)`
Dirichlet	**Densità**	`ddirichlet(x=c(`$x_1,\ldots,x_k$`),alpha=c(`$\alpha_1,\ldots,\alpha_k$`))`
	Estrazione random	`rdirichlet(n,alpha=c(`$\alpha_1,\ldots,\alpha_k$`))`
Normale doppia	**Densità**	`dmvnorm(x=c(`x_1,x_2`),mean=c(`μ_1,μ_2`),sigma=`V`)`
	Ripartizione	`pmvnorm(upper=c(`x_1,x_2`),mean=c(`μ_1,μ_2`),sigma=`V`)`
	Estrazione random	`rmvnorm(n,mean=c(`μ_1,μ_2`),sigma=`V`)`
Normale multipla	**Densità**	`dmvnorm(x=c(`$x_1,\ldots,x_k$`),mean=c(`$\mu_1,\ldots,\mu_k$`),sigma=`V`)`
	Ripartizione	`pmvnorm(upper=c(`$x_1,\ldots,x_k$`),mean=c(`$\mu_1,\ldots,\mu_k$`),sigma=`V`)`
	Estrazione random	`rmvnorm(n,mean=c(`$\mu_1,\ldots,\mu_k$`),sigma=`V`)`

3.13 Funzioni ai valori mancanti

is.na()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** rileva la presenza di valori `NA` oppure `NaN`
- **Esempio:**

```
> x
[1] 1.3 1.0 2.0 3.4 3.4 5.7  NA 3.8
> is.na(x)
[1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE

> x
[1] 1.3 NaN 2.0 3.4 3.4 5.7  NA 3.8
> is.na(x)
[1] FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE

> x
[1] 1.0 2.0  NA 4.0 5.6 NaN 1.2 4.0 4.4
> x[!is.na(x)]
[1] 1.0 2.0 4.0 5.6 1.2 4.0 4.4
> # x privato dei valori NA e NaN

> x
[1]  3  4 NA  5
> mean(x)
[1] NA
> mean(x[!is.na(x)])
[1] 4
```

is.nan()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Significato:** rileva la presenza di valori `NaN`
- **Esempio:**

```
> x
[1] 1.3 1.0 2.0 3.4 3.4 5.7  NA 3.8
> is.nan(x)
[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE

> x
[1] 1.3 NaN 2.0 3.4 3.4 5.7  NA 3.8
> is.nan(x)
[1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE

> x
[1] 1.0 2.0  NA 4.0 5.6 NaN 1.2 4.0 4.4
> x[!is.nan(x)]
[1] 1.0 2.0  NA 4.0 5.6 1.2 4.0 4.4
> # x privato dei valori NaN
```

3.14 Miscellaneous

ic.var()

- **Package:** `labstatR`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `conf.level` livello di confidenza $1-\alpha$

- **Significato:** intervallo di confidenza Chi-Quadrato per la varianza incognita
- **Formula:**

$$\frac{(n-1)\,s_x^2}{\chi^2_{1-\alpha/2,\,n-1}} \quad \frac{(n-1)\,s_x^2}{\chi^2_{\alpha/2,\,n-1}}$$

- **Esempio:**

```
> x
[1]  1.2  3.4  4.2 12.4 13.4 17.3 18.1
> n<-length(x)
> n
[1] 7
> alpha<-0.05
> lower<-(n-1)*var(x)/qchisq(1-alpha/2,df=n-1)
> upper<-(n-1)*var(x)/qchisq(alpha/2,df=n-1)
> c(lower,upper)
[1]  20.12959 235.06797
> ic.var(x,conf.level=0.95)
[1]  20.12959 235.06797

> x
[1] 1.0 2.0 3.0 4.0 5.6 7.4 1.2 4.0 4.4
> n<-length(x)
> n
```

```
[1] 9
> alpha<-0.05
> lower<-(n-1)*var(x)/qchisq(1-alpha/2,df=n-1)
> upper<-(n-1)*var(x)/qchisq(alpha/2,df=n-1)
> c(lower,upper)
[1]  1.986681 15.981587
> ic.var(x,conf.level=0.95)
[1]  1.986681 15.981587
```

sample()

- **Package:** `base`
- **Parametri:**

  `x` vettore alfanumerico di dimensione n

  `size` ampiezza campionaria

  `replace = T / F` estrazione con oppure senza ripetizione

  `prob` vettore di probabilità

- **Significato:** estrazione campionaria
- **Esempio:**

```
> x<-c("A","B")
> n<-length(x)
> n
[1] 2
> sample(x,size=10,replace=T,prob=rep(1/n,n))
[1] "A" "A" "A" "B" "A" "B" "A" "A" "B" "B"

> x<-c(0,1)
> n<-length(x)
> n
[1] 2
> sample(x,size=5,replace=T,prob=rep(1/n,n))
[1] 0 1 0 0 0

> x<-c(1,2,3,4,5,6,7,8,9,10)
> n<-length(x)
> n
[1] 10
> sample(x,size=3,replace=F,prob=rep(1/n,n))
[1] 6 8 4
```

is.finite()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** valore numerico finito
- **Esempio:**

```
> x<-2.3
> is.finite(x)
[1] TRUE
```

```
> x<-1/0
> is.finite(x)
[1] FALSE

> x<-0/0
> is.finite(x)
[1] FALSE
```

is.infinite()

- **Package:** `base`
- **Parametri:**

  `x` valore numerico

- **Significato:** valore numerico infinito
- **Esempio:**

```
> x<-2.3
> is.infinite(x)
[1] FALSE

> x<-1/0
> is.infinite(x)
[1] TRUE

> x<-0/0
> is.infinite(x)
[1] FALSE
```

diff()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n
  `lag` il valore d del ritardo
  `differences` il valore k dell'ordine delle differenze

- **Significato:** differenze in una serie storica
- **Formula:**

$$\left(1 - B^d\right)^k x_t \quad \forall t = d\,k + 1,\, d\,k + 2,\, \ldots,\, n$$

$$\text{dove} \quad B^h\, x_t = x_{t-h} \quad \forall t = h + 1,\, h + 2,\, \ldots,\, n$$

- **Esempio:**

```
> x
[1]  1  2  4  3  5  6 -9
> n<-length(x)
> n
[1] 7
> k<-1
> d<-2
> x[-(1:d)]-x[-((n-d+1):n)]
[1]   3   1   1   3 -14
> diff(x,lag=d,differences=k)
[1]   3   1   1   3 -14
```

scale()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico di dimensione n
  `center = T / F` parametro di posizione
  `scale = T / F` parametro di scala

- **Significato:** centratura o normalizzazione
- **Formula:**

	`scale = T`	`scale = F`
`center = T`	$(x - \bar{x}) / s_x$	$x - \bar{x}$
`center = F`	$x / \left(\frac{1}{n-1} \sum_{i=1}^{n} x_i^2\right)^{1/2}$	x

- **Esempio:**

```
> x
[1]  1.2  3.4  4.2 12.4 13.4 17.3 18.1
> n<-length(x)
> n
[1] 7
> (x-mean(x))/sd(x)
[1] -1.264 -0.948 -0.833  0.345  0.488  1.048  1.163
> as.numeric(scale(x,center=T,scale=T))
[1] -1.264 -0.948 -0.833  0.345  0.488  1.048  1.163
> x-mean(x)
[1] -8.8 -6.6 -5.8  2.4  3.4  7.3  8.1
> as.numeric(scale(x,center=T,scale=F))
[1] -8.8 -6.6 -5.8  2.4  3.4  7.3  8.1
> x/sqrt(sum(x**2)/(length(x)-1))
[1] 0.0934 0.2646 0.3268 0.9649 1.0427 1.3462 1.4085
> as.numeric(scale(x,center=F,scale=T))
[1] 0.0934 0.2646 0.3268 0.9649 1.0427 1.3462 1.4085
> x
[1]  1.2  3.4  4.2 12.4 13.4 17.3 18.1
> as.numeric(scale(x,center=F,scale=F))
[1]  1.2  3.4  4.2 12.4 13.4 17.3 18.1

> x
[1] 1.2 4.5 6.7 7.8 9.8
> n<-length(x)
> n
[1] 5
> (x-mean(x))/sd(x)
[1] -1.4562179 -0.4550681  0.2123651  0.5460817  1.1528392
> as.numeric(scale(x,center=T,scale=T))
[1] -1.4562179 -0.4550681  0.2123651  0.5460817  1.1528392
> x-mean(x)
[1] -4.8 -1.5  0.7  1.8  3.8
> as.numeric(scale(x,center=T,scale=F))
[1] -4.8 -1.5  0.7  1.8  3.8
> x/sqrt(sum(x**2)/(length(x)-1))
[1] 0.1605504 0.6020639 0.8964063 1.0435775 1.3111615
> as.numeric(scale(x,center=F,scale=T))
[1] 0.1605504 0.6020639 0.8964063 1.0435775 1.3111615
> x
[1] 1.2 4.5 6.7 7.8 9.8
> as.numeric(scale(x,center=F,scale=F))
[1] 1.2 4.5 6.7 7.8 9.8
```

moment()

- **Package:** `e1071`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `center = T / F` parametro di posizione

  `absolute = T / F` modulo

  `order` il valore k dell'ordine

- **Significato:** momento centrato e non centrato di ordine k
- **Formula:**

	`absolute = T`	`absolute = F`
`center = T`	$\sum_{i=1}^{n} \lvert x_i - \bar{x}\rvert^k / n$	$\sum_{i=1}^{n} (x_i - \bar{x})^k / n$
`center = F`	$\sum_{i=1}^{n} \lvert x_i\rvert^k / n$	$\sum_{i=1}^{n} x_i^k / n$

- **Esempio:**

```
> x
[1] -1.2  1.2  3.4  4.2 12.4 13.4 17.3 18.1
> n<-length(x)
> n
[1] 8
> k<-5
> mean(abs(x-mean(x))**k)
[1] 31074.24
> moment(x,center=T,absolute=T,order=k)
[1] 31074.24
> mean((x-mean(x))**k)
[1] 1565.904
> moment(x,center=T,absolute=F,order=k)
[1] 1565.904
> mean(abs(x)**k)
[1] 527406.3
> moment(x,center=F,absolute=T,order=k)
[1] 527406.3
> mean(x**k)
[1] 527405.6
> moment(x,center=F,absolute=F,order=k)
[1] 527405.6

> x
[1] 1.2 4.5 6.7 7.8 9.8
> n<-length(x)
> n
[1] 5
> k<-3
> mean(abs(x-mean(x))**k)
[1] 35.0028
> moment(x,center=T,absolute=T,order=k)
[1] 35.0028
> mean((x-mean(x))**k)
[1] -10.584
> moment(x,center=T,absolute=F,order=k)
[1] -10.584
> mean(abs(x)**k)
[1] 361.872
> moment(x,center=F,absolute=T,order=k)
[1] 361.872
> mean(x**k)
```

```
[1] 361.872
> moment(x,center=F,absolute=F,order=k)
[1] 361.872
```

cum3()

- **Package:** `boot`
- **Parametri:**

  `a` vettore numerico x di dimensione n

  `b` vettore numerico y di dimensione n

  `c` vettore numerico z di dimensione n

  `unbiased = T / F` distorsione

- **Significato:** momento terzo centrato
- **Formula:**

`unbiased = T`

$$\frac{n}{(n-1)(n-2)} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})(z_i - \bar{z})$$

`unbiased = F`

$$\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})(z_i - \bar{z})$$

- **Esempio:**

```
> x
[1] -3 -2 -1  0  1  2
> y
[1] 1.20 2.30 2.00 3.10 3.55 6.70
> z
[1] 2.00 3.45 2.60 3.11 3.50 6.20
> n<-length(x)
> n
[1] 6
> (n/((n-1)*(n-2)))*sum((x-mean(x))*(y-mean(y))*(z-mean(z)))
[1] 4.96385
> cum3(a=x,b=y,c=z,unbiased=T)
[1] 4.96385

> x
[1] -3 -2 -1  0  1  2
> y
[1] 1.20 2.30 2.00 3.10 3.55 6.70
> z
[1] 2.00 3.45 2.60 3.11 3.50 6.20
> n<-length(x)
> n
[1] 6
> (1/n)*sum((x-mean(x))*(y-mean(y))*(z-mean(z)))
[1] 2.757694
> cum3(a=x,b=y,c=z,unbiased=F)
[1] 2.757694
```

sweep()

- **Package:** `base`
- **Parametri:**

  `x` matrice di dimensione $n \times k$

  `MARGIN = 1 / 2` righe oppure colonne

  `STATS` statistica da sottrarre da ogni riga o colonna della matrice x

- **Significato:** sottrae $STATS$ da ogni riga o colonna della matrice x
- **Esempio:**

```
> x
      X1  X2  X3
[1,] 1.2 7.5 4.3
[2,] 3.4 6.7 3.2
[3,] 5.6 8.4 3.2
> X1media<-mean(x[,"X1"])
> X2media<-mean(x[,"X2"])
> X3media<-mean(x[,"X3"])
> mediecolonna<-c(X1media,X2media,X3media)
> mediecolonna
[1] 3.400000 7.533333 3.566667
> prova<-sweep(x,MARGIN=2,STATS=mediecolonna)
> prova
       X1          X2         X3
[1,] -2.2 -0.03333333  0.7333333
[2,]  0.0 -0.83333333 -0.3666667
[3,]  2.2  0.86666667 -0.3666667
> # verifica
> apply(prova,MARGIN=2,FUN=mean)
           X1            X2            X3
-1.480297e-16 -5.921189e-16  2.960595e-16
> # verifica OK

> x
      X1  X2  X3
[1,] 1.2 7.5 4.3
[2,] 3.4 6.7 3.2
[3,] 5.6 8.4 3.2
> X1mediana<-median(x[,"X1"])
> X2mediana<-median(x[,"X2"])
> X3mediana<-median(x[,"X3"])
> medianecolonna<-c(X1mediana,X2mediana,X3mediana)
> medianecolonna
[1] 3.4 7.5 3.2
> prova<-sweep(x,MARGIN=2,STATS=medianecolonna)
> prova
       X1   X2  X3
[1,] -2.2  0.0 1.1
[2,]  0.0 -0.8 0.0
[3,]  2.2  0.9 0.0
> # verifica
> apply(prova,MARGIN=2,FUN=median)
X1 X2 X3
 0  0  0
> # verifica OK
```

nsize()

- **Package:** `BSDA`
- **Parametri:**

  `b` valore del margine di errore E

  `sigma` valore dello scarto quadratico medio σ_x

  `p` valore della proporzione campionaria p

  `conf.level` livello di confidenza $1 - \alpha$

  `type = mu / pi` media nella popolazione oppure proporzione campionaria

- **Significato:** dimensione campionaria dato il margine di errore E
- **Formula:**

`type = mu`

$$n = \lceil (z_{1-\alpha\,/\,2}\,\sigma_x)\,/\,E)^2 \rceil$$

`type = pi`

$$n = \lceil p\,(1-p)\,(z_{1-\alpha\,/\,2}\,/\,E)^2 \rceil$$

- **Esempio:**

```
> nsize(b=0.15,sigma=0.31,conf.level=0.95,type="mu")

The required sample size (n) to estimate the population mean with a
0.95 confidence interval so that the margin of error is no more than
0.15 is 17 .

> nsize(b=0.03,p=0.77,conf.level=0.95,type="pi")

The required sample size (n) to estimate the population proportion
of successes with a 0.95 confidence interval so that the margin of
error is no more than 0.03 is 756 .
```

Capitolo 4

Analisi Componenti Principali (ACP)

4.1 ACP con matrice di covarianza

Simbologia

- matrice dei dati di dimensione $n \times k$ le cui colonne corrispondono ai vettori numerici $w_1, w_2, \ldots, w_k$: W
- media di colonna della matrice dei dati: $\bar{w}_j \quad \forall j = 1, 2, \ldots, k$
- matrice centrata di dimensione $n \times k$: Z
- elemento di riga i e colonna j della matrice centrata:
  $z_{ij} = w_{ij} - \bar{w}_j \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$
- matrice di covarianza di dimensione $k \times k$: $S = \frac{Z^T Z}{n-1} = \Gamma D \Gamma^T$
- matrice ortogonale degli autovettori di dimensione $k \times k$: Γ
- j-esima colonna della matrice Γ: $\Gamma^j \quad \forall j = 1, 2, \ldots, k$
- matrice diagonale degli autovalori di dimensione $k \times k$: $D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_k)$
- componente principale j-esima: $x_j = Z \Gamma^j \quad \forall j = 1, 2, \ldots, k$
- deviazione standard della j-esima componente principale:
  $s_{x_j} = \sqrt{\lambda_{(k-j+1)}} \quad \forall j = 1, 2, \ldots, k$
- problema di ottimo vincolato:
  $x_j = Z \gamma_j \quad \forall j = 1, 2, \ldots, k$
  $s^2_{x_j} = \frac{x_j^T x_j}{n-1} = \frac{(Z \gamma_j)^T (Z \gamma_j)}{n-1} = \gamma_j^T \frac{Z^T Z}{n-1} \gamma_j = \gamma_j^T S \gamma_j \quad \forall j = 1, 2, \ldots, k$
  $\max_{\gamma_j^T \gamma_j = 1} s^2_{x_j} = \max_{\gamma_j^T \gamma_j = 1} \gamma_j^T S \gamma_j = \lambda_{(k-j+1)} \quad \forall j = 1, 2, \ldots, k$

prcomp()

- **Parametri:**

  `W` matrice dei dati

- **Output:**

  `sdev` deviazione standard delle componenti principali

  `rotation` matrice ortogonale degli autovettori

  `center` media di colonna della matrice W

  `x` componenti principali

- **Formula:**

  `sdev`

  $$s_{x_j} \quad \forall j = 1, 2, \ldots, k$$

  `rotation`

  $$\Gamma$$

`center`

$$\bar{w}_j \quad \forall\, j = 1, 2, \ldots, k$$

`x`

$$x_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Esempio:**

```
> n<-nrow(W)
> k<-ncol(W)
> Z<-scale(W,scale=F)
> S<-1/(n-1)*t(Z)%*%Z
> D<-diag(eigen(S)$values)
> GAMMA<-eigen(S)$vectors
> prcomp(W,scale=F) # ACP con matrice di covarianza
```

summary()

- **Parametri:**

  `object` oggetto di tipo `prcomp()`

- **Output:**

  `sdev` deviazione standard delle componenti principali

  `rotation` matrice ortogonale degli autovettori

  `center` media di colonna della matrice W

  `x` componenti principali

  `importance` deviazione standard delle componenti principali, quota di varianza spiegata da ciascuna componente principale e le quota di varianza spiegata dalle prime l componenti principali ($l = 1, 2, \ldots, k$)

- **Formula:**

  `sdev`

  $$s_{x_j} \quad \forall\, j = 1, 2, \ldots, k$$

  `rotation`

  $$\Gamma$$

  `center`

  $$\bar{w}_j \quad \forall\, j = 1, 2, \ldots, k$$

  `x`

  $$x_j \quad \forall\, j = 1, 2, \ldots, k$$

  `importance`

  $$s_{x_j} \qquad \frac{\lambda_{(k-j+1)}}{\sum_{i=1}^{k} \lambda_i} \qquad \frac{\sum_{j=1}^{l} \lambda_{(k-j+1)}}{\sum_{i=1}^{k} \lambda_i} \qquad \forall\, j, l = 1, 2, \ldots, k$$

- **Esempio:**

```
> summary(object=prcomp(W,scale=F)) # ACP con matrice di covarianza
```

4.2 ACP con matrice di correlazione

Simbologia

- matrice dei dati di dimensione $n \times k$ le cui colonne corrispondono ai vettori numerici $w_1, w_2, \ldots, w_k$: W
- media di colonna della matrice dei dati: $\bar{w}_j \quad \forall j = 1, 2, \ldots, k$
- varianza campionaria di colonna della matrice dei dati:
  $s^2_{w_j} = (n-1)^{-1} (w_j - \bar{w}_j)^T (w_j - \bar{w}_j) \quad \forall j = 1, 2, \ldots, k$
- matrice standardizzata di dimensione $n \times k$: Z
- elemento di riga i e colonna j della matrice standardizzata:
  $z_{ij} = (w_{ij} - \bar{w}_j) / s_{w_j} \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$
- matrice di correlazione di dimensione $k \times k$: $R = \frac{Z^T Z}{n-1} = \Gamma D \Gamma^T$
- matrice ortogonale degli autovettori di dimensione $k \times k$: Γ
- j-esima colonna della matrice Γ: $\Gamma^j \quad \forall j = 1, 2, \ldots, k$
- matrice diagonale degli autovalori di dimensione $k \times k$: $D = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_k)$
- componente principale j-esima: $x_j = Z \Gamma^j \quad \forall j = 1, 2, \ldots, k$
- deviazione standard della j-esima componente principale:
  $s_{x_j} = \sqrt{\lambda_{(k-j+1)}} \quad \forall j = 1, 2, \ldots, k$
- problema di ottimo vincolato:
  $x_j = Z \gamma_j \quad \forall j = 1, 2, \ldots, k$
  $s^2_{x_j} = \frac{x_j^T x_j}{n-1} = \frac{(Z\gamma_j)^T (Z\gamma_j)}{n-1} = \gamma_j^T \frac{Z^T Z}{n-1} \gamma_j = \gamma_j^T R \gamma_j \quad \forall j = 1, 2, \ldots, k$
  $\max_{\gamma_j^T \gamma_j = 1} s^2_{x_j} = \max_{\gamma_j^T \gamma_j = 1} \gamma_j^T R \gamma_j = \lambda_{(k-j+1)} \quad \forall j = 1, 2, \ldots, k$

prcomp()

- **Parametri:**

  `W` matrice dei dati

- **Output:**

  `sdev` deviazione standard delle componenti principali

  `rotation` matrice ortogonale degli autovettori

  `center` media di colonna della matrice W

  `scale` deviazione standard di colonna della matrice W

  `x` componenti principali

- **Formula:**

  `sdev`
  $$s_{x_j} \quad \forall j = 1, 2, \ldots, k$$

  `rotation`
  $$\Gamma$$

  `center`
  $$\bar{w}_j \quad \forall j = 1, 2, \ldots, k$$

  `scale`
  $$s_{w_j} \quad \forall j = 1, 2, \ldots, k$$

  `x`
  $$x_j \quad \forall j = 1, 2, \ldots, k$$

- **Esempio:**

```
> n<-nrow(W)
> k<-ncol(W)
> Z<-scale(W)
> R<-1/(n-1)*t(Z)%*%Z
> D<-diag(eigen(R)$values)
> GAMMA<-eigen(R)$vectors
> prcomp(W,scale=T) # ACP con matrice di correlazione
```

summary()

- **Parametri:**

  `object` oggetto di tipo `prcomp()`

- **Output:**

  `sdev` deviazione standard delle componenti principali

  `rotation` matrice ortogonale degli autovettori

  `center` media di colonna della matrice W

  `scale` deviazione standard di colonna della matrice W

  `x` componenti principali

  `importance` deviazione standard delle componenti principali, quota di varianza spiegata da ciascuna componente principale e le quota di varianza spiegata dalle prime l componenti principali ($l = 1, 2, \ldots, k$)

- **Formula:**

  `sdev`

  $$s_{x_j} \quad \forall\, j = 1, 2, \ldots, k$$

  `rotation`

  $$\Gamma$$

  `center`

  $$\bar{w}_j \quad \forall\, j = 1, 2, \ldots, k$$

  `scale`

  $$s_{w_j} \quad \forall\, j = 1, 2, \ldots, k$$

  `x`

  $$x_j \quad \forall\, j = 1, 2, \ldots, k$$

  `importance`

  $$s_{x_j} \qquad \frac{\lambda_{(k-j+1)}}{k} \qquad \frac{1}{k}\sum_{j=1}^{l} \lambda_{(k-j+1)} \qquad \forall\, j, l = 1, 2, \ldots, k$$

- **Esempio:**

```
> summary(object=prcomp(W,scale=T)) # ACP con matrice di correlazione
```

Capitolo 5

Analisi dei Gruppi

5.1 Indici di distanza

dist()

- **Package:** `stats`
- **Parametri:**

  `x` matrice di dimensione $n \times k$ le cui righe corrispondono ai vettori numerici x_1, x_2, …, x_n

  `method = euclidean / maximum / manhattan / canberra / binary / minkowski` indice di distanza

  `p` valore p di potenza per la distanza di *Minkowski*

- **Significato:** matrice di distanza o di dissimilarità per gli n vettori di dimensione $n \times n$
- **Formula:**

`method = euclidean`

$$d_{x_i x_j} = \left(\sum_{h=1}^{k} (x_{ih} - x_{jh})^2 \right)^{1/2} \qquad \forall\, i, j = 1, 2, \ldots, n$$

`method = maximum`

$$d_{x_i x_j} = \max_h |x_{ih} - x_{jh}| \quad \forall\, i, j = 1, 2, \ldots, n$$

`method = manhattan`

$$d_{x_i x_j} = \sum_{h=1}^{k} |x_{ih} - x_{jh}| \quad \forall\, i, j = 1, 2, \ldots, n$$

`method = canberra`

$$d_{x_i x_j} = \sum_{h=1}^{k} \frac{x_{ih} - x_{jh}}{x_{ih} + x_{jh}} \quad \forall\, i, j = 1, 2, \ldots, n$$

`method = binary`

$$d_{x_i x_j} = 1 - \frac{n_{11}}{n_{01} + n_{10} + n_{11}} \quad \forall\, i, j = 1, 2, \ldots, n$$

`method = minkowski`

$$d_{x_i x_j} = \left(\sum_{h=1}^{k} |x_{ih} - x_{jh}|^p \right)^{1/p} \qquad \forall\, i, j = 1, 2, \ldots, n$$

- **Esempio:**

```
> x<-matrix(rnorm(30),nrow=10,ncol=3,byrow=F)
> k<-3
> n<-10
> dist(x,method="euclidean",upper=T,diag=T)
> dist(x,method="minkowski",p=1,upper=T,diag=T)
```

- **Osservazioni 1:** Possiamo ottenere le variabili standardizzate se applichiamo il comando scale() alla matrice `x`.
- **Osservazioni 2:** La distanza di dissimilarità calcolata con `method = binary` corrisponde al complemento ad uno dell'indice di *Jaccard*.

as.dist()

- **Package:** `stats`
- **Parametri:**

  `m` matrice simmetrica con elementi nulli sulla diagonale di dimensione $n \times n$

  `upper = T / F` matrice triangolare superiore

  `diag = T / F` elementi nulli sulla diagonale

- **Significato:** oggetto di tipo `dist()`
- **Esempio:**

```
> m
     [,1] [,2] [,3]
[1,]    0    1    5
[2,]    1    0    3
[3,]    5    3    0
> n<-3
> as.dist(m,upper=T,diag=T)
  1 2 3
1 0 1 5
2 1 0 3
3 5 3 0
> as.dist(m,upper=T,diag=F)
  1 2 3
1   1 5
2 1   3
3 5 3
> as.dist(m,upper=F,diag=T)
  1 2 3
1 0
2 1 0
3 5 3 0
> as.dist(m,upper=F,diag=F)
  1 2
2 1
3 5 3
```

5.2 Criteri di Raggruppamento

hclust()

- **Package:** `stats`
- **Parametri:**

`d` oggetto di tipo `dist()`

`method = ward / single / complete / average / mcquitty / median / centroid` criterio di *Ward*, *Legame Singolo*, *Legame Completo*, *Legame Medio*, *McQuitty*, *Mediana* e *Centroide*

- **Significato:** analisi dei gruppi per gli n vettori di dimensione k
- **Output:**

`merge` matrice di dimensione $(n-1) \times 2$ le cui righe descrivono le aggregazioni avvenute a ciascun passo dell'intero procedimento. Gli elementi negativi indicano singole unità, mentre quelli positivi indicano gruppi già formati

`height` vettore di $n-1$ valori numerici non decrescenti che indicano i livelli di dissomiglianza ai quali avvengono le aggregazioni

`order` permutazioni delle osservazioni originali

`order` vettore delle etichette delle osservazioni

`method` criterio di aggregazione utilizzato

`dist.method` criterio di distanza utilizzato

- **Formula:**

`method = ward`

$$d_{(xy)z} = \frac{(n_x + n_z)\, d_{xz} + (n_y + n_z)\, d_{yz} - n_z\, d_{(xy)}}{n_{xy} + n_z}$$

`method = single`

$$d_{(xy)z} = \min(d_{xz},\, d_{yz})$$

`method = complete`

$$d_{(xy)z} = \max(d_{xz},\, d_{yz})$$

`method = average`

$$d_{(xy)z} = \frac{n_x\, d_{xz} + n_y\, d_{yz}}{n_{(xy)}}$$

`method = mcquitty`

$$d_{(xy)z} = \frac{d_{xz} + d_{yz}}{2}$$

`method = median`

$$d_{(xy)z} = \frac{d_{xz} + d_{yz}}{2} - \frac{d_{(xy)}}{4}$$

`method = centroid`

$$d_{(xy)z} = \frac{n_x\, d_{xz} + n_y\, d_{yz}}{n_{(xy)}} - \frac{n_x\, n_y\, d_{xy}}{n_{(xy)}^2}$$

- **Esempio:**

```
> x<-matrix(rnorm(30),nrow=3,ncol=10,byrow=F)
> k<-3
> n<-10
> d<-dist(x,method="euclidean",upper=T,diag=T)
> hclust(d=d,method="single")
```

kmeans()

- **Package:** `stats`
- **Parametri:**

  `x` matrice di dimensione $n \times k$ le cui righe corrispondono ai vettori numerici $x_1, x_2, \ldots, x_n$

  `centers` scalare che indica il numero di gruppi

  `iter.max` massimo numero di iterazioni permesse per l'ottimizzazione

- **Significato:** analisi di ragguppamento non gerarchica con il metodo *k-means*
- **Output:**

  `cluster` vettore di numeri interi che indicano l'appartenenza di ciascuna osservazione ad uno dei gruppi individuati

  `centers` centroidi dei gruppi ottenuti

  `withinss` devianza di ciascun gruppo

  `size` numero di osservazioni in ciascun gruppo

- **Esempio:**

```
> x<-rbind(matrix(rnorm(100,sd=0.3),ncol=2)
> kmeans(x,centers=2,iter.max=10)
```

Parte III

Statistica Inferenziale

Capitolo 6

Test di ipotesi parametrici

6.1 Test di ipotesi sulla media con uno o due campioni

Test Z con un campione

- **Package:** `BSDA`
- **Sintassi:** `z.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `sigma.x` valore di σ_x

  `mu` valore di μ_0

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value

  `conf.int` intervallo di confidenza per la media incognita a livello $1 - \alpha$

  `estimate` media campionaria

  `null.value` valore di μ_0

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

  $$z = \frac{\bar{x} - \mu_0}{\sigma_x \,/\, \sqrt{n}}$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\,\lvert z \rvert)$

  `conf.int`

  $$\bar{x} \mp z_{1-\alpha\,/\,2}\,\sigma_x \,/\, \sqrt{n}$$

  `estimate`

  $$\bar{x}$$

  `null.value`

  $$\mu_0$$

- **Esempio:**

```
> x
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1 6.7 7.6 6.8
> xmedio<-mean(x)
> xmedio
[1] 7.018182
> sigmax<-1.2
> n<-length(x)
> n
[1] 11
> mu0<-6.5
> z<-(xmedio-mu0)/(sigmax/sqrt(n))
> z
[1] 1.432179
> res<-z.test(x,sigma.x=1.2,mu=6.5,alternative="two.sided",conf.level=0.95)
> res$statistic
       z
1.432179
> p.value<-2*pnorm(-abs(z))
> p.value
[1] 0.1520926
> res$p.value
[1] 0.1520926
> alpha<-0.05
> lower<-xmedio-qnorm(1-0.05/2)*sigmax/sqrt(n)
> upper<-xmedio+qnorm(1-0.05/2)*sigmax/sqrt(n)
> c(lower,upper)
[1] 6.309040 7.727323
> res$conf.int
[1] 6.309040 7.727323
attr(,"conf.level")
[1] 0.95
> xmedio
[1] 7.018182
> res$estimate
mean of x
 7.018182
> mu0
[1] 6.5
> res$null.value
mean
 6.5
> res$alternative
[1] "two.sided"

> x
[1] 1.0 2.3 4.5 6.7 8.9
> xmedio<-mean(x)
> xmedio
[1] 4.68
> sigmax<-1.45
> n<-length(x)
> n
[1] 5
> mu0<-5.2
> z<-(xmedio-mu0)/(sigmax/sqrt(n))
> z
[1] -0.8019002
> res<-z.test(x,sigma.x=1.45,mu=5.2,alternative="two.sided",conf.level=0.95)
> res$statistic
         z
-0.8019002
```

```
> p.value<-2*pnorm(-abs(z))
> p.value
[1] 0.4226107
> res$p.value
[1] 0.4226107
> alpha<-0.05
> lower<-xmedio-qnorm(1-0.05/2)*sigmax/sqrt(n)
> upper<-xmedio+qnorm(1-0.05/2)*sigmax/sqrt(n)
> c(lower,upper)
[1] 3.409042 5.950958
> res$conf.int
[1] 3.409042 5.950958
attr(,"conf.level")
[1] 0.95
> xmedio
[1] 4.68
> res$estimate
mean of x
     4.68
> mu0
[1] 5.2
> res$null.value
mean
 5.2
> res$alternative
[1] "two.sided"
```

Test di Student con un campione

- **Package:** `stats`
- **Sintassi:** `t.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `mu` valore di μ_0

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica t

  `parameter` gradi di libertà

  `p.value` p-value

  `conf.int` intervallo di confidenza per la media incognita a livello $1 - \alpha$

  `estimate` media campionaria

  `null.value` valore di μ_0

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$t = \frac{\bar{x} - \mu_0}{s_x / \sqrt{n}}$$

  `parameter`

$$df = n - 1$$

`p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\,P(t_{df} \leq -\lvert t \rvert)$

`conf.int`

$$\bar{x} \mp t_{1-\alpha\,/\,2,\,df}\; s_x \,/\, \sqrt{n}$$

`estimate`

$$\bar{x}$$

`null.value`

$$\mu_0$$

- **Esempio:**

```
> x
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1 6.7 7.6 6.8
> xmedio<-mean(x)
> xmedio
[1] 7.018182
> sx<-sd(x)
> sx
[1] 0.4643666
> n<-length(x)
> n
[1] 11
> mu0<-6.5
> t<-(xmedio-mu0)/(sx/sqrt(n))
> t
[1] 3.700987
> res<-t.test(x,mu=6.5,alternative="two.sided",conf.level=0.95)
> res$statistic
       t
3.700987
> parameter<-n-1
> parameter
[1] 10
> res$parameter
    df
[1] 10
> p.value<-2*pt(-abs(t),df=n-1)
> p.value
[1] 0.004101817
> res$p.value
[1] 0.004101817
> alpha<-0.05
> lower<-xmedio-qt(1-0.05/2,df=n-1)*sx/sqrt(n)
> upper<-xmedio+qt(1-0.05/2,df=n-1)*sx/sqrt(n)
> c(lower,upper)
[1] 6.706216 7.330148
> res$conf.int
[1] 6.706216 7.330148
attr(,"conf.level")
[1] 0.95
> xmedio
[1] 7.018182
> res$estimate
mean of x
 7.018182
> mu0
[1] 6.5
> res$null.value
mean
```

```
 6.5
> res$alternative
[1] "two.sided"

> x
[1] 1.0 2.3 4.5 6.7 8.9
> xmedio<-mean(x)
> xmedio
[1] 4.68
> sx<-sd(x)
> sx
[1] 3.206556
> n<-length(x)
> n
[1] 5
> mu0<-5.2
> t<-(xmedio-mu0)/(sx/sqrt(n))
> t
[1] -0.3626182
> res<-t.test(x,mu=5.2,alternative="two.sided",conf.level=0.95)
> res$statistic
         t
-0.3626182
> parameter<-n-1
> parameter
[1] 4
> res$parameter
   df
[1] 4
> p.value<-2*pt(-abs(t),df=n-1)
> p.value
[1] 0.7352382
> res$p.value
[1] 0.7352382
> alpha<-0.05
> lower<-xmedio-qt(1-0.05/2,df=n-1)*sx/sqrt(n)
> upper<-xmedio+qt(1-0.05/2,df=n-1)*sx/sqrt(n)
> c(lower,upper)
[1] 0.6985351 8.6614649
> res$conf.int
[1] 0.6985351 8.6614649
attr(,"conf.level")
[1] 0.95
> mean(x)
[1] 4.68
> res$estimate
mean of x
     4.68
> mu0
[1] 5.2
> res$null.value
mean
 5.2
> res$alternative
[1] "two.sided"
```

Test Z con due campioni indipendenti

- **Package:** `BSDA`
- **Sintassi:** `z.test()`

- **Parametri:**

  `x` vettore numerico di dimensione n_x

  `y` vettore numerico di dimensione n_y

  `sigma.x` valore di σ_x

  `sigma.y` valore di σ_y

  `mu` valore di $(\mu_x - \mu_y)_{|H_0}$

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value

  `conf.int` intervallo di confidenza per la differenza tra le medie incognite a livello $1 - \alpha$

  `estimate` medie campionarie

  `null.value` valore di $(\mu_x - \mu_y)_{|H_0}$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$z = \frac{(\bar{x} - \bar{y}) - (\mu_x - \mu_y)_{|H_0}}{\sqrt{\sigma_x^2 / n_x + \sigma_y^2 / n_y}}$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\lvert z \rvert)$

  `conf.int`

$$\bar{x} - \bar{y} \mp z_{1-\alpha/2} \sqrt{\sigma_x^2 / n_x + \sigma_y^2 / n_y}$$

  `estimate`

$$\bar{x} \qquad \bar{y}$$

  `null.value`

$$(\mu_x - \mu_y)_{|H_0}$$

- **Esempio:**

```
> x
[1] 154 109 137 115 140
> xmedio<-mean(x)
> xmedio
[1] 131
> sigmax<-15.5
> nx<-length(x)
> nx
[1] 5
> y
[1] 108 115 126  92 146
> ymedio<-mean(y)
> ymedio
[1] 117.4
> sigmay<-13.5
> ny<-length(y)
> ny
[1] 5
> mu0<-10
> z<-(xmedio-ymedio-mu0)/sqrt(sigmax**2/nx+sigmay**2/ny)
> z
```

```
[1] 0.3916284
> res<-z.test(x,y,sigma.x=15.5,sigma.y=13.5,mu=10,
+ alternative="two.sided",conf.level=0.95)
> res$statistic
        z
0.3916284
> p.value<-2*pnorm(-abs(z))
> p.value
[1] 0.6953328
> res$p.value
[1] 0.6953328
> alpha<-0.05
> lower<-(xmedio-ymedio)-qnorm(1-0.05/2)*sqrt(sigmax**2/nx+sigmay**2/ny)
> upper<-(xmedio-ymedio)+qnorm(1-0.05/2)*sqrt(sigmax**2/nx+sigmay**2/ny)
> c(lower,upper)
[1] -4.41675 31.61675
> res$conf.int
[1] -4.41675 31.61675
attr(,"conf.level")
[1] 0.95
> c(xmedio,ymedio)
[1] 131.0 117.4
> res$estimate
mean of x mean of y
    131.0     117.4
> mu0
[1] 10
> res$null.value
difference in means
                 10
> res$alternative
[1] "two.sided"

> x
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1 6.7 7.6 6.8
> xmedio<-mean(x)
> xmedio
[1] 7.018182
> sigmax<-0.5
> nx<-length(x)
> nx
[1] 11
> y
[1] 4.5 5.4 6.1 6.1 5.4 5.0 4.1 5.5
> ymedio<-mean(y)
> ymedio
[1] 5.2625
> sigmay<-0.8
> ny<-length(y)
> ny
[1] 8
> mu0<-1.2
> z<-(xmedio-ymedio-mu0)/sqrt(sigmax**2/nx+sigmay**2/ny)
> z
[1] 1.733737
> res<-z.test(x,y,sigma.x=0.5,sigma.y=0.8,mu=1.2,
+ alternative="two.sided",conf.level=0.95)
> res$statistic
       z
1.733737
> p.value<-2*pnorm(-abs(z))
> p.value
[1] 0.0829647
```

```
> res$p.value
[1] 0.0829647
> alpha<-0.05
> lower<-(xmedio-ymedio)-qnorm(1-0.05/2)*sqrt(sigmax**2/nx+sigmay**2/ny)
> upper<-(xmedio-ymedio)+qnorm(1-0.05/2)*sqrt(sigmax**2/nx+sigmay**2/ny)
> c(lower,upper)
[1] 1.127492 2.383872
> res$conf.int
[1] 1.127492 2.383872
attr(,"conf.level")
[1] 0.95
> c(xmedio,ymedio)
[1] 7.018182 5.262500
> res$estimate
mean of x mean of y
 7.018182  5.262500
> mu0
[1] 1.2
> res$null.value
difference in means
                1.2
> res$alternative
[1] "two.sided"
```

Test di Student con due campioni indipendenti con varianze non note ma supposte uguali

- **Package:** `stats`
- **Sintassi:** `t.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n_x

  `y` vettore numerico di dimensione n_y

  `mu` valore di $(\mu_x - \mu_y)_{|H_0}$

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica t

  `parameter` gradi di libertà

  `p.value` p-value

  `conf.int` intervallo di confidenza per la differenza tra le medie incognite a livello $1 - \alpha$

  `estimate` medie campionarie

  `null.value` valore di $(\mu_x - \mu_y)_{|H_0}$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$t = \frac{(\bar{x} - \bar{y}) - (\mu_x - \mu_y)_{|H_0}}{s_P \sqrt{1/n_x + 1/n_y}}$$

$$\text{dove} \quad s_P^2 = \frac{(n_x - 1)\, s_x^2 + (n_y - 1)\, s_y^2}{n_x + n_y - 2}$$

  `parameter`

$$df = n_x + n_y - 2$$

p.value

alternative	less	greater	two.sided
p.value	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\,P(t_{df} \leq -\lvert t \rvert)$

conf.int

$$\bar{x} - \bar{y} \mp t_{1-\alpha\,/\,2,\,df}\; s_P \sqrt{1\,/\,n_x + 1\,/\,n_y}$$

estimate

$$\bar{x} \qquad \bar{y}$$

null.value

$$(\,\mu_x - \mu_y\,)_{|\,H_0}$$

- **Esempio:**

```
> x
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1 6.7 7.6 6.8
> xmedio<-mean(x)
> xmedio
[1] 7.018182
> sx<-sd(x)
> sx
[1] 0.4643666
> nx<-length(x)
> nx
[1] 11
> y
[1] 4.5 5.4 6.1 6.1 5.4 5.0 4.1 5.5
> ymedio<-mean(y)
> ymedio
[1] 5.2625
> sy<-sd(y)
> sy
[1] 0.7069805
> ny<-length(y)
> ny
[1] 8
> mu0<-1.2
> Sp<-sqrt(((nx-1)*sx**2+(ny-1)*sy**2)/(nx+ny-2))
> Sp
[1] 0.5767614
> t<-(xmedio-ymedio-mu0)/(Sp*sqrt(1/nx+1/ny))
> t
[1] 2.073455
> res<-t.test(x,y,mu=1.2,alternative="two.sided",conf.level=0.95,var.equal=T)
> res$statistic
       t
2.073455
> parameter<-nx+ny-2
> parameter
[1] 17
> res$parameter
   df
[1] 17
> p.value<-2*pt(-abs(t),df=nx+ny-2)
> p.value
[1] 0.05364043
> res$p.value
[1] 0.05364043
> alpha<-0.05
> lower<-(xmedio-ymedio)-qt(1-0.05/2,df=nx+ny-2)*Sp*sqrt(1/nx+1/ny)
> upper<-(xmedio-ymedio)+qt(1-0.05/2,df=nx+ny-2)*Sp*sqrt(1/nx+1/ny)
> c(lower,upper)
```

```
[1] 1.190255 2.321108
> res$conf.int
[1] 1.190255 2.321108
attr(,"conf.level")
[1] 0.95
> c(xmedio,ymedio)
[1] 7.018182 5.262500
> res$estimate
mean of x mean of y
 7.018182  5.262500
> mu0
[1] 1.2
> res$null.value
difference in means
                1.2
> res$alternative
[1] "two.sided"

> x
[1] 154 109 137 115 140
> xmedio<-mean(x)
> xmedio
[1] 131
> sx<-sd(x)
> sx
[1] 18.61451
> nx<-length(x)
> nx
[1] 5
> y
[1] 108 115 126  92 146
> ymedio<-mean(y)
> ymedio
[1] 117.4
> sy<-sd(y)
> sy
[1] 20.19406
> ny<-length(y)
> ny
[1] 5
> mu0<-10
> Sp<-sqrt(((nx-1)*sx**2+(ny-1)*sy**2)/(nx+ny-2))
> Sp
[1] 19.42035
> t<-(xmedio-ymedio-mu0)/(Sp*sqrt(1/nx+1/ny))
> t
[1] 0.2930998
> res<-t.test(x,y,mu=10,alternative="two.sided",conf.level=0.95,var.equal=T)
> res$statistic
        t
0.2930998
> parameter<-nx+ny-2
> parameter
[1] 8
> res$parameter
   df
[1] 8
> p.value<-2*pt(-abs(t),df=nx+ny-2)
> p.value
[1] 0.7769049
> res$p.value
[1] 0.7769049
> alpha<-0.05
```

```
> lower<-(xmedio-ymedio)-qt(1-0.05/2,df=nx+ny-2)*Sp*sqrt(1/nx+1/ny)
> upper<-(xmedio-ymedio)+qt(1-0.05/2,df=nx+ny-2)*Sp*sqrt(1/nx+1/ny)
> c(lower,upper)
[1] -14.72351  41.92351
> res$conf.int
[1] -14.72351  41.92351
attr(,"conf.level")
[1] 0.95
> c(xmedio,ymedio)
[1] 131.0 117.4
> res$estimate
mean of x mean of y
    131.0     117.4
> mu0
[1] 10
> res$null.value
difference in means
                 10
> res$alternative
[1] "two.sided"
```

Test di Student con due campioni indipendenti con varianze non note e diverse

- **Package:** `stats`
- **Sintassi:** `t.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n_x

  `y` vettore numerico di dimensione n_y

  `mu` valore di $(\mu_x - \mu_y)_{|H_0}$

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1-\alpha$

- **Output:**

  `statistic` valore empirico della statistica t

  `parameter` gradi di libertà

  `p.value` p-value

  `conf.int` intervallo di confidenza per la differenza tra le medie incognite a livello $1-\alpha$

  `estimate` medie campionarie

  `null.value` valore di $(\mu_x - \mu_y)_{|H_0}$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$t = \frac{(\bar{x}-\bar{y}) - (\mu_x - \mu_y)_{|H_0}}{\sqrt{s_x^2 / n_x + s_y^2 / n_y}}$$

  `parameter`

$$df = \frac{\left(s_x^2 / n_x + s_y^2 / n_y\right)^2}{s_x^4 / (n_x^2 (n_x - 1)) + s_y^4 / (n_y^2 (n_y - 1))}$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\,P(t_{df} \leq -\lvert t \rvert)$

`conf.int`

$$\bar{x} - \bar{y} \mp t_{1-\alpha\,/\,2,\,df}\,\sqrt{s_x^2\,/\,n_x + s_y^2\,/\,n_y}$$

`estimate`

$$\bar{x} \qquad \bar{y}$$

`null.value`

$$(\,\mu_x - \mu_y\,)_{|\,H_0}$$

- **Esempio:**

```
> x
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1 6.7 7.6 6.8
> xmedio<-mean(x)
> xmedio
[1] 7.018182
> sx<-sd(x)
> sx
[1] 0.4643666
> nx<-length(x)
> nx
[1] 11
> y
[1] 4.5 5.4 6.1 6.1 5.4 5.0 4.1 5.5
> ymedio<-mean(y)
> ymedio
[1] 5.2625
> sy<-sd(y)
> sy
[1] 0.7069805
> ny<-length(y)
> ny
[1] 8
> mu0<-1.2
> t<-(xmedio-ymedio-mu0)/sqrt(sx**2/nx+sy**2/ny)
> t
[1] 1.939568
> res<-t.test(x,y,mu=1.2,alternative="two.sided",conf.level=0.95,var.equal=F)
> res$statistic
       t
1.939568
> gl<-(sx**2/nx+sy**2/ny)**2/(sx**4/(nx**2*(nx-1))+sy**4/(ny**2*(ny-1)))
> gl
[1] 11.30292
> res$parameter
[1] 11.30292
> p.value<-2*pt(-abs(t),df=gl)
> p.value
[1] 0.07779219
> res$p.value
[1] 0.07779219
> lower<-(xmedio-ymedio)-qt(1-0.05/2,df=gl)*sqrt(sx**2/nx+sy**2/ny)
> upper<-(xmedio-ymedio)+qt(1-0.05/2,df=gl)*sqrt(sx**2/nx+sy**2/ny)
> c(lower,upper)
[1] 1.127160 2.384203
> res$conf.int
[1] 1.127160 2.384203
attr(,"conf.level")
[1] 0.95
> c(xmedio,ymedio)
[1] 7.018182 5.262500
> res$estimate
mean of x mean of y
 7.018182  5.262500
```

```
> mu0
[1] 1.2
> res$null.value
difference in means
                1.2
> res$alternative
[1] "two.sided"

> x
[1] 154 109 137 115 140
> xmedio<-mean(x)
> xmedio
[1] 131
> sx<-sd(x)
> sx
[1] 18.61451
> nx<-length(x)
> nx
[1] 5
> y
[1] 108 115 126  92 146
> ymedio<-mean(y)
> ymedio
[1] 117.4
> sy<-sd(y)
> sy
[1] 20.19406
> ny<-length(y)
> ny
[1] 5
> mu0<-10
> t<-(xmedio-ymedio-mu0)/sqrt(sx**2/nx+sy**2/ny)
> t
[1] 0.2930998
> res<-t.test(x,y,mu=10,alternative="two.sided",conf.level=0.95,var.equal=F)
> res$statistic
        t
0.2930998
> gl<-(sx**2/nx+sy**2/ny)**2/(sx**4/(nx**2*(nx-1))+sy**4/(ny**2*(ny-1)))
> gl
[1] 7.947512
> res$parameter
          df
[1] 7.947512
> p.value<-2*pt(-abs(t),df=gl)
> p.value
[1] 0.7769531
> res$p.value
[1] 0.7769531
> alpha<-0.05
> lower<-(xmedio-ymedio)-qt(1-0.05/2,df=gl)*sqrt(sx**2/nx+sy**2/ny)
> upper<-(xmedio-ymedio)+qt(1-0.05/2,df=gl)*sqrt(sx**2/nx+sy**2/ny)
> c(lower,upper)
[1] -14.75611  41.95611
> res$conf.int
[1] -14.75611  41.95611
attr(,"conf.level")
[1] 0.95
> c(xmedio,ymedio)
[1] 131.0 117.4
> res$estimate
mean of x mean of y
    131.0     117.4
```

```
> mu0
[1] 10
> res$null.value
difference in means
                 10
> res$alternative
[1] "two.sided"
```

Test di Student per dati appaiati

- **Package:** `stats`
- **Sintassi:** `t.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `y` vettore numerico di dimensione n

  `mu` valore di $(\mu_x - \mu_y)_{|H_0}$

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica t

  `parameter` gradi di libertà

  `p.value` p-value

  `conf.int` intervallo di confidenza per la differenza tra le medie incognite a livello $1 - \alpha$

  `estimate` differenza tra le medie campionarie

  `null.value` valore di $(\mu_x - \mu_y)_{|H_0}$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$t = \frac{(\bar{x} - \bar{y}) - (\mu_x - \mu_y)_{|H_0}}{s_{x-y} / \sqrt{n}}$$

$$\text{dove} \quad s^2_{x-y} = \frac{1}{n-1} \sum_{i=1}^{n} \left(x_i - y_i - (\bar{x} - \bar{y})\right)^2$$

  `parameter`

$$df = n - 1$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\,P(t_{df} \leq -\lvert t \rvert)$

  `conf.int`

$$\bar{x} - \bar{y} \mp t_{1-\alpha/2,\,df}\, s_{x-y} / \sqrt{n}$$

  `estimate`

$$\bar{x} - \bar{y}$$

  `null.value`

$$(\mu_x - \mu_y)_{|H_0}$$

- **Esempio:**

```
> x
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1
> xmedio<-mean(x)
> xmedio
[1] 7.0125
> y
[1] 4.5 5.4 6.1 6.1 5.4 5.0 4.1 5.5
> ymedio<-mean(y)
> ymedio
[1] 5.2625
> n<-length(x)
> n
[1] 8
> mu0<-1.2
> t<-(xmedio-ymedio-mu0)/(sd(x-y)/sqrt(n))
> t
[1] 1.815412
> res<-t.test(x,y,mu=1.2,alternative="two.sided",conf.level=0.95,paired=T)
> res$statistic
       t
1.815412
> parameter<-n-1
> parameter
[1] 7
> res$parameter
   df
[1] 7
> p.value<-2*pt(-abs(t),df=n-1)
> p.value
[1] 0.1123210
> res$p.value
[1] 0.1123210
> alpha<-0.05
> lower<-(xmedio-ymedio)-qt(1-0.05/2,df=n-1)*sd(x-y)/sqrt(n)
> upper<-(xmedio-ymedio)+qt(1-0.05/2,df=n-1)*sd(x-y)/sqrt(n)
> c(lower,upper)
[1] 1.033610 2.466390
> res$conf.int
[1] 1.033610 2.466390
attr(,"conf.level")
[1] 0.95
> xmedio-ymedio
[1] 1.75
> res$estimate
mean of the differences
                  1.75
> mu0
[1] 1.2
> res$null.value
difference in means
                1.2
> res$alternative
[1] "two.sided"

> x
[1] 154 109 137 115 140
> xmedio<-mean(x)
> xmedio
[1] 131
> y
[1] 108 115 126  92 146
```

```
> ymedio<-mean(y)
> ymedio
[1] 117.4
> n<-length(x)
> n
[1] 5
> mu0<-10
> t<-(xmedio-ymedio-mu0)/(sd(x-y)/sqrt(n))
> t
[1] 0.3680758
> res<-t.test(x,y,mu=10,alternative="two.sided",conf.level=0.95,paired=T)
> res$statistic
        t
0.3680758
> parameter<-n-1
> parameter
[1] 4
> res$parameter
   df
[1] 4
> p.value<-2*pt(-abs(t),df=n-1)
> p.value
[1] 0.7314674
> res$p.value
[1] 0.7314674
> alpha<-0.05
> lower<-(xmedio-ymedio)-qt(1-0.05/2,df=n-1)*sd(x-y)/sqrt(n)
> upper<-(xmedio-ymedio)+qt(1-0.05/2,df=n-1)*sd(x-y)/sqrt(n)
> c(lower,upper)
[1] -13.55528  40.75528
> res$conf.int
[1] -13.55528  40.75528
attr(,"conf.level")
[1] 0.95
> xmedio-ymedio
[1] 13.6
> res$estimate
mean of the differences
                   13.6
> mu0
[1] 10
> res$null.value
difference in means
                 10
> res$alternative
[1] "two.sided"
```

6.2 Test di ipotesi sulla media con uno o due campioni (summarized data)

Test Z con un campione

- **Package:** `BSDA`
- **Sintassi:** `zsum.test()`
- **Parametri:**

  `mean.x` valore di $\bar{x}$

  `sigma.x` valore di σ_x

  `n.x` valore di n

`mu` valore di μ_0

`alternative = less / greater / two.sided` ipotesi alternativa

`conf.level` livello di confidenza $1 - \alpha$

- **Output:**

`statistic` valore empirico della statistica Z

`p.value` p-value

`conf.int` intervallo di confidenza per la media incognita a livello $1 - \alpha$

`estimate` media campionaria

`null.value` valore di μ_0

`alternative` ipotesi alternativa

- **Formula:**

`statistic`

$$z = \frac{\bar{x} - \mu_0}{\sigma_x \,/\, \sqrt{n}}$$

`p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\,\lvert z \rvert)$

`conf.int`

$$\bar{x} \mp z_{1-\alpha\,/\,2}\,\sigma_x \,/\, \sqrt{n}$$

`estimate`

$$\bar{x}$$

`null.value`

$$\mu_0$$

- **Esempio:**

```
> xmedio<-7.018182
> sigmax<-1.2
> n<-11
> mu0<-6.5
> z<-(xmedio-mu0)/(sigmax/sqrt(n))
> z
[1] 1.432179
> res<-zsum.test(mean.x=7.018182,sigma.x=1.2,n.x=11,mu=6.5,
+ alternative="two.sided",conf.level=0.95)
> res$statistic
       z
1.432179
> p.value<-2*pnorm(-abs(z))
> p.value
[1] 0.1520926
> res$p.value
[1] 0.1520926
> alpha<-0.05
> lower<-xmedio-qnorm(1-0.05/2)*sigmax/sqrt(n)
> upper<-xmedio+qnorm(1-0.05/2)*sigmax/sqrt(n)
> c(lower,upper)
[1] 6.309040 7.727323
> res$conf.int
[1] 6.309040 7.727323
attr(,"conf.level")
[1] 0.95
> xmedio
[1] 7.018182
> res$estimate
```

```
mean of x
 7.018182
> mu0
[1] 6.5
> res$null.value
mean
 6.5
> res$alternative
[1] "two.sided"

> xmedio<-4.68
> sigmax<-1.45
> n<-5
> mu0<-5.2
> z<-(xmedio-mu0)/(sigmax/sqrt(n))
> z
[1] -0.8019002
> res<-zsum.test(mean.x=4.68,sigma.x=1.45,n.x=5,mu=5.2,
+ alternative="two.sided",conf.level=0.95)
> res$statistic
         z
-0.8019002
> p.value<-2*pnorm(-abs(z))
> p.value
[1] 0.4226107
> res$p.value
[1] 0.4226107
> alpha<-0.05
> lower<-xmedio-qnorm(1-0.05/2)*sigmax/sqrt(n)
> upper<-xmedio+qnorm(1-0.05/2)*sigmax/sqrt(n)
> c(lower,upper)
[1] 3.409042 5.950958
> res$conf.int
[1] 3.409042 5.950958
attr(,"conf.level")
[1] 0.95
> xmedio
[1] 4.68
> res$estimate
mean of x
     4.68
> mu0
[1] 5.2
> res$null.value
mean
 5.2
> res$alternative
[1] "two.sided"
```

Test di Student con un campione

- **Package:** `BSDA`
- **Sintassi:** `tsum.test()`
- **Parametri:**

  `mean.x` valore di $\bar{x}$

  `s.x` valore di s_x

  `n.x` valore di n

  `mu` valore di μ_0

`alternative = less / greater / two.sided` ipotesi alternativa

`conf.level` livello di confidenza $1-\alpha$

- **Output:**

`statistic` valore empirico della statistica t

`parameter` gradi di libertà

`p.value` p-value

`conf.int` intervallo di confidenza per la media incognita a livello $1-\alpha$

`estimate` media campionaria

`null.value` valore di μ_0

`alternative` ipotesi alternativa

- **Formula:**

`statistic`

$$t = \frac{\bar{x} - \mu_0}{s_x / \sqrt{n}}$$

`parameter`

$$df = n - 1$$

`p.value`

alternative	less	greater	two.sided
p.value	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\,P(t_{df} \leq -\lvert t \rvert)$

`conf.int`

$$\bar{x} \mp t_{1-\alpha/2,\,df}\, s_x / \sqrt{n}$$

`estimate`

$$\bar{x}$$

`null.value`

$$\mu_0$$

- **Esempio:**

```
> xmedio<-7.018182
> sx<-1.2
> n<-11
> mu0<-6.5
> t<-(xmedio-mu0)/(sx/sqrt(n))
> t
[1] 1.432179
> res<-tsum.test(mean.x=7.018182,s.x=1.2,n.x=11,
+ mu=6.5,alternative="two.sided",conf.level=.95)
> res$statistic
       t
1.432179
> parameter<-n-1
> parameter
[1] 10
> res$parameter
   df
[1] 10
> p.value<-2*pt(-abs(t),df=n-1)
> p.value
[1] 0.1826001
> res$p.value
[1] 0.1826001
> alpha<-0.05
> lower<-xmedio-qt(1-0.05/2,df=n-1)*sx/sqrt(n)
> upper<-xmedio+qt(1-0.05/2,df=n-1)*sx/sqrt(n)
```

```
> c(lower,upper)
[1] 6.212011 7.824353
> res$conf.int
[1] 6.212011 7.824353
attr(,"conf.level")
[1] 0.95
> xmedio
[1] 7.018182
> res$estimate
mean of x
 7.018182
> mu0
[1] 6.5
> res$null.value
mean
 6.5
> res$alternative
[1] "two.sided"

> xmedio<-4.68
> sx<-1.45
> n<-5
> mu0<-5.2
> t<-(xmedio-mu0)/(sx/sqrt(n))
> t
[1] -0.8019002
> res<-tsum.test(mean.x=4.68,s.x=1.45,n.x=5,
+ mu=5.2,alternative="two.sided",conf.level=.95)
> res$statistic
         t
-0.8019002
> parameter<-n-1
> parameter
[1] 4
> res$parameter
   df
[1] 4
> p.value<-2*pt(-abs(t),df=n-1)
> p.value
[1] 0.4675446
> res$p.value
[1] 0.4675446
> alpha<-0.05
> lower<-xmedio-qt(1-0.05/2,df=n-1)*sx/sqrt(n)
> upper<-xmedio+qt(1-0.05/2,df=n-1)*sx/sqrt(n)
> c(lower,upper)
[1] 2.879587 6.480413
> res$conf.int
[1] 2.879587 6.480413
attr(,"conf.level")
[1] 0.95
> xmedio
[1] 4.68
> res$estimate
mean of x
     4.68
> mu0
[1] 5.2
> res$null.value
mean
 5.2
> res$alternative
[1] "two.sided"
```

Test Z con due campioni indipendenti

- **Package:** `BSDA`
- **Sintassi:** `zsum.test()`
- **Parametri:**

  `mean.x` valore di $\bar{x}$

  `sigma.x` valore di σ_x

  `n.x` valore di n_x

  `mean.y` valore di $\bar{y}$

  `sigma.y` valore di σ_y

  `n.y` valore di n_y

  `mu` valore di $(\,\mu_x - \mu_y\,)_{|\,H_0}$

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value

  `conf.int` intervallo di confidenza per la differenza tra le medie incognite a livello $1 - \alpha$

  `estimate` medie campionarie

  `null.value` valore di $(\,\mu_x - \mu_y\,)_{|\,H_0}$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

  $$z \;=\; \frac{(\bar{x} - \bar{y}) - (\,\mu_x - \mu_y\,)_{|\,H_0}}{\sqrt{\sigma_x^2 \,/\, n_x + \sigma_y^2 \,/\, n_y}}$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\,|\,z\,|)$

  `conf.int`

  $$\bar{x} - \bar{y} \mp z_{1-\alpha\,/\,2}\,\sqrt{\sigma_x^2 \,/\, n_x + \sigma_y^2 \,/\, n_y}$$

  `estimate`

  $$\bar{x} \qquad \bar{y}$$

  `null.value`

  $$(\,\mu_x - \mu_y\,)_{|\,H_0}$$

- **Esempio:**

```
> xmedio<-131
> sigmax<-15.5
> nx<-5
> ymedio<-117.4
> sigmay<-13.5
> ny<-5
> mu0<-10
> z<-(xmedio-ymedio-mu0)/sqrt(sigmax**2/nx+sigmay**2/ny)
> z
[1] 0.3916284
> res<-zsum.test(mean.x=131,sigma.x=15.5,n.x=5,mean.y=117.4,sigma.y=13.5,n.y=5,
```

```
+ mu=10,alternative="two.sided",conf.level=0.95)
> res$statistic
        z
0.3916284
> p.value<-2*pnorm(-abs(z))
> p.value
[1] 0.6953328
> res$p.value
[1] 0.6953328
> alpha<-0.05
> lower<-xmedio-ymedio-qnorm(1-0.05/2)*sqrt(sigmax**2/nx+sigmay**2/ny)
> upper<-xmedio-ymedio+qnorm(1-0.05/2)*sqrt(sigmax**2/nx+sigmay**2/ny)
> c(lower,upper)
[1] -4.41675 31.61675
> res$conf.int
[1] -4.41675 31.61675
attr(,"conf.level")
[1] 0.95
> c(xmedio,ymedio)
[1] 131.0 117.4
> res$estimate
mean of x mean of y
    131.0     117.4
> mu0
[1] 10
> res$null.value
difference in means
                 10
> res$alternative
[1] "two.sided"

> xmedio<-7.018182
> sigmax<-0.5
> nx<-11
> ymedio<-5.2625
> sigmay<-0.8
> ny<-8
> mu0<-1.2
> z<-(xmedio-ymedio-mu0)/sqrt(sigmax**2/nx+sigmay**2/ny)
> z
[1] 1.733738
> res<-zsum.test(mean.x=7.018182,sigma.x=0.5,n.x=11,mean.y=5.2625,
+ sigma.y=0.8,n.y=8,mu=1.2,alternative="two.sided",conf.level=0.95)
> res$statistic
       z
1.733738
> p.value<-2*pnorm(-abs(z))
> p.value
[1] 0.0829646
> res$p.value
[1] 0.0829646
> alpha<-0.05
> lower<-xmedio-ymedio-qnorm(1-0.05/2)*sqrt(sigmax**2/nx+sigmay**2/ny)
> upper<-xmedio-ymedio+qnorm(1-0.05/2)*sqrt(sigmax**2/nx+sigmay**2/ny)
> c(lower,upper)
[1] 1.127492 2.383872
> res$conf.int
[1] 1.127492 2.383872
attr(,"conf.level")
[1] 0.95
> c(xmedio,ymedio)
[1] 7.018182 5.262500
> res$estimate
```

```
mean of x mean of y
 7.018182  5.262500
> mu0
[1] 1.2
> res$null.value
difference in means
                1.2
> res$alternative
[1] "two.sided"
```

Test di Student con due campioni indipendenti con varianze non note ma supposte uguali

- **Package:** `BSDA`
- **Sintassi:** `tsum.test()`
- **Parametri:**

  `mean.x` valore di $\bar{x}$

  `s.x` valore di s_x

  `n.x` valore di n_x

  `mean.y` valore di $\bar{y}$

  `s.y` valore di s_y

  `n.y` valore di n_y

  `mu` valore di $(\mu_x - \mu_y)_{|H_0}$

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica t

  `parameter` gradi di libertà

  `p.value` p-value

  `conf.int` intervallo di confidenza per la differenza tra le medie incognite a livello $1 - \alpha$

  `estimate` medie campionarie

  `null.value` valore di $(\mu_x - \mu_y)_{|H_0}$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$t = \frac{(\bar{x} - \bar{y}) - (\mu_x - \mu_y)_{|H_0}}{s_P \sqrt{1/n_x + 1/n_y}}$$

$$\text{dove} \quad s_P^2 = \frac{(n_x - 1)\, s_x^2 + (n_y - 1)\, s_y^2}{n_x + n_y - 2}$$

  `parameter`

$$df = n_x + n_y - 2$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\,P(t_{df} \leq -\lvert t \rvert)$

  `conf.int`

$$\bar{x} - \bar{y} \mp t_{1-\alpha/2,\, df}\, s_P \sqrt{1/n_x + 1/n_y}$$

`estimate`

$$\bar{x} \qquad \bar{y}$$

`null.value`

$$(\,\mu_x - \mu_y\,)_{|\,H_0}$$

- **Esempio:**

```
> xmedio<-7.018182
> sx<-0.5
> nx<-11
> ymedio<-5.2625
> sy<-0.8
> ny<-8
> mu0<-1.2
> Sp<-sqrt(((nx-1)*sx**2+(ny-1)*sy**2)/(nx+ny-2))
> Sp
[1] 0.6407716
> t<-(xmedio-ymedio-mu0)/(Sp*sqrt(1/nx+1/ny))
> t
[1] 1.866326
> res<-tsum.test(mean.x=7.018182,s.x=0.5,n.x=11,mean.y=5.2625,s.y=0.8,n.y=8,
+ mu0<-1.2,alternative="two.sided",conf.level=0.95)
> res$statistic
       t
1.866326
> parameter<-nx+ny-2
> parameter
[1] 17
> res$parameter
    df
[1] 17
> p.value<-2*pt(-abs(t),df=nx+ny-2)
> p.value
[1] 0.07934364
> res$p.value
[1] 0.07934364
> alpha<-0.05
> lower<-(xmedio-ymedio)-qt(1-0.05/2,df=nx+ny-2)*Sp*sqrt(1/nx+1/ny)
> upper<-(xmedio-ymedio)+qt(1-0.05/2,df=nx+ny-2)*Sp*sqrt(1/nx+1/ny)
> c(lower,upper)
[1] 1.127503 2.383861
> res$conf.int
[1] 1.127503 2.383861
attr(,"conf.level")
[1] 0.95
> c(xmedio,ymedio)
[1] 7.018182 5.262500
> res$estimate
mean of x mean of y
 7.018182  5.262500
> mu0
[1] 1.2
> res$null.value
difference in means
                1.2
> res$alternative
[1] "two.sided"

> xmedio<-131
> sx<-15.5
> nx<-5
> ymedio<-117.4
> sy<-13.5
```

```
> ny<-5
> mu0<-10
> Sp<-sqrt(((nx-1)*sx**2+(ny-1)*sy**2)/(nx+ny-2))
> Sp
[1] 14.53444
> t<-(xmedio-ymedio-mu0)/(Sp*sqrt(1/nx+1/ny))
> t
[1] 0.3916284
> res<-tsum.test(mean.x=131,s.x=15.5,n.x=5,mean.y=117.4,s.y=13.5,n.y=5,
+ mu=10,alternative="two.sided",conf.level=0.95,var.equal=T)
> res$statistic
        t
0.3916284
> parameter<-nx+ny-2
> parameter
[1] 8
> res$parameter
   df
[1] 8
> p.value<-2*pt(-abs(t),df=nx+ny-2)
> p.value
[1] 0.705558
> res$p.value
[1] 0.705558
> alpha<-0.05
> lower<-(xmedio-ymedio)-qt(1-0.05/2,df=nx+ny-2)*Sp*sqrt(1/nx+1/ny)
> upper<-(xmedio-ymedio)+qt(1-0.05/2,df=nx+ny-2)*Sp*sqrt(1/nx+1/ny)
> c(lower,upper)
[1] -7.597685 34.797685
> res$conf.int
[1] -7.597685 34.797685
attr(,"conf.level")
[1] 0.95
> c(xmedio,ymedio)
[1] 131.0 117.4
> res$estimate
mean of x mean of y
    131.0     117.4
> mu0
[1] 10
> res$null.value
difference in means
                 10
> res$alternative
[1] "two.sided"
```

Test di Student con due campioni indipendenti con varianze non note e diverse

- **Package:** `BSDA`
- **Sintassi:** `tsum.test()`
- **Parametri:**

  `mean.x` valore di $\bar{x}$

  `s.x` valore di s_x

  `n.x` valore di n_x

  `mean.y` valore di $\bar{y}$

  `s.y` valore di s_y

  `n.y` valore di n_y

  `mu` valore di $(\mu_x - \mu_y)_{|H_0}$

`alternative = less / greater / two.sided` ipotesi alternativa

`conf.level` livello di confidenza $1-\alpha$

- **Output:**

`statistic` valore empirico della statistica t

`parameter` gradi di libertà

`p.value` p-value

`conf.int` intervallo di confidenza per la differenza tra le medie incognite a livello $1-\alpha$

`estimate` medie campionarie

`null.value` valore di $(\mu_x - \mu_y)_{|H_0}$

`alternative` ipotesi alternativa

- **Formula:**

`statistic`

$$t = \frac{(\bar{x}-\bar{y}) - (\mu_x - \mu_y)_{|H_0}}{\sqrt{s_x^2 / n_x + s_y^2 / n_y}}$$

`parameter`

$$df = \frac{\left(s_x^2 / n_x + s_y^2 / n_y\right)^2}{s_x^4 / (n_x^2 (n_x - 1)) + s_y^4 / (n_y^2 (n_y - 1))}$$

`p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\,P(t_{df} \leq -\lvert t \rvert)$

`conf.int`

$$\bar{x} - \bar{y} \mp t_{1-\alpha/2,\,df} \sqrt{s_x^2 / n_x + s_y^2 / n_y}$$

`estimate`

$$\bar{x} \qquad \bar{y}$$

`null.value`

$$(\mu_x - \mu_y)_{|H_0}$$

- **Esempio:**

```
> xmedio<-7.018182
> sx<-0.5
> nx<-11
> ymedio<-5.2625
> sy<-0.8
> ny<-8
> mu0<-1.2
> t<-(xmedio-ymedio-mu0)/sqrt(sx**2/nx+sy**2/ny)
> t
[1] 1.733738
> res<-tsum.test(mean.x=7.018182,s.x=0.5,n.x=11,mean.y=5.2625,s.y=0.8,n.y=8,
+ mu=1.2,alternative="two.sided",conf.level=0.95,var.equal=F)
> res$statistic
       t
1.733738
> gl<-(sx**2/nx+sy**2/ny)**2/(sx**4/(nx**2*(nx-1))+sy**4/(ny**2*(ny-1)))
> gl
[1] 10.92501
> res$parameter
[1] 10.92501
> p.value<-2*pt(-abs(t),df=gl)
> p.value
[1] 0.1110536
> res$p.value
```

```
[1] 0.1110536
> lower<-(xmedio-ymedio)-qt(1-0.05/2,df=gl)*sqrt(sx**2/nx+sy**2/ny)
> upper<-(xmedio-ymedio)+qt(1-0.05/2,df=gl)*sqrt(sx**2/nx+sy**2/ny)
> c(lower,upper)
[1] 1.049651 2.461713
> res$conf.int
[1] 1.049651 2.461713
attr(,"conf.level")
[1] 0.95
> c(xmedio,ymedio)
[1] 7.018182 5.262500
> res$estimate
mean of x mean of y
 7.018182  5.262500
> mu0
[1] 1.2
> res$null.value
difference in means
                1.2
> res$alternative
[1] "two.sided"

> xmedio<-131
> sx<-15.5
> nx<-5
> ymedio<-117.4
> sy<-13.5
> ny<-5
> mu0<-10
> t<-(xmedio-ymedio-mu0)/sqrt(sx**2/nx+sy**2/ny)
> t
[1] 0.3916284
> res<-tsum.test(mean.x=131,s.x=15.5,n.x=5,mean.y=117.4,s.y=13.5,n.y=5,
+ mu=10,alternative="two.sided",conf.level=0.95,var.equal=F)
> res$statistic
        t
0.3916284
> gl<-(sx**2/nx+sy**2/ny)**2/(sx**4/(nx**2*(nx-1))+sy**4/(ny**2*(ny-1)))
> gl
[1] 7.852026
> res$parameter
          df
[1] 7.852026
> p.value<-2*pt(-abs(t),df=gl)
> p.value
[1] 0.7057463
> res$p.value
[1] 0.7057463
> lower<-(xmedio-ymedio)-qt(1-0.05/2,df=gl)*sqrt(sx**2/nx+sy**2/ny)
> upper<-(xmedio-ymedio)+qt(1-0.05/2,df=gl)*sqrt(sx**2/nx+sy**2/ny)
> c(lower,upper)
[1] -7.667421 34.867421
> res$conf.int
[1] -7.667421 34.867421
attr(,"conf.level")
[1] 0.95
> c(xmedio,ymedio)
[1] 131.0 117.4
> res$estimate
mean of x mean of y
    131.0     117.4
> mu0
[1] 10
```

```
> res$null.value
difference in means
                 10
> res$alternative
[1] "two.sided"
```

6.3 Test di ipotesi sulla varianza con uno o due campioni

Test Chi-Quadrato con un campione

- **Package:** `sigma2tools`
- **Sintassi:** `sigma2.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `var0` valore di σ_0^2

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

  `conf.int` intervallo di confidenza per la media incognita a livello $1 - \alpha$

  `estimate` varianza campionaria

  `null.value` valore di σ_0^2

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

  $$c = \frac{(n-1)\, s_x^2}{\sigma_0^2}$$

  `parameter`

  $$df = n - 1$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$P(\chi^2_{df} \leq c)$	$P(\chi^2_{df} \geq c)$	$2 \min\Big(P(\chi^2_{df} \leq c),\, P(\chi^2_{df} \geq c)\Big)$

  `conf.int`

  $$\frac{(n-1)\, s_x^2}{\chi^2_{1-\alpha\,/\,2,\,df}} \quad \frac{(n-1)\, s_x^2}{\chi^2_{\alpha\,/\,2,\,df}}$$

  `estimate`

  $$s_x^2$$

  `null.value`

  $$\sigma_0^2$$

- **Esempio:**

```
> x
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1 6.7 7.6 6.8
> sx<-sd(x)
> sx
[1] 0.4643666
> n<-length(x)
> n
[1] 11
> var0<-0.5
> c<-(n-1)*sx**2/var0
> c
[1] 4.312727
> res<-sigma2.test(x,var0=0.5,alternative="two.sided",conf.level=0.95)
> res$statistic
X-squared
 4.312727
> parameter<-n-1
> parameter
[1] 10
> res$parameter
df
10
> p.value<-2*min(pchisq(c,df=n-1),1-pchisq(c,df=n-1))
> p.value
[1] 0.1357229
> res$p.value
[1] 0.1357229
> alpha<-0.05
> lower<-(n-1)*sx**2/qchisq(1-alpha/2,df=n-1)
> upper<-(n-1)*sx**2/qchisq(alpha/2,df=n-1)
> c(lower,upper)
[1] 0.1052749 0.6641151
> res$conf.int
[1] 0.1052749 0.6641151
attr(,"conf.level")
[1] 0.95
> sx**2
[1] 0.2156364
> res$estimate
 var of x
0.2156364
> var0
[1] 0.5
> res$null.value
variance
     0.5
> res$alternative
[1] "two.sided"

> x
[1] 1.0 2.3 4.5 6.7 8.9
> sx<-sd(x)
> sx
[1] 3.206556
> n<-length(x)
> n
[1] 5
> var0<-12
> c<-(n-1)*sx**2/var0
> c
[1] 3.427333
```

```
> res<-sigma2.test(x,var0=12,alternative="two.sided",conf.level=0.95)
> res$statistic
X-squared
 3.427333
> parameter<-n-1
> parameter
[1] 4
> res$parameter
df
 4
> p.value<-2*min(pchisq(c,df=n-1),1-pchisq(c,df=n-1))
> p.value
[1] 0.9780263
> res$p.value
[1] 0.9780263
> alpha<-0.05
> lower<-(n-1)*sx**2/qchisq(1-alpha/2,df=n-1)
> upper<-(n-1)*sx**2/qchisq(alpha/2,df=n-1)
> c(lower,upper)
[1]  3.690832 84.901785
> res$conf.int
[1]  3.690832 84.901785
attr(,"conf.level")
[1] 0.95
> sx**2
[1] 10.282
> res$estimate
var of x
  10.282
> var0
[1] 12
> res$null.value
variance
      12
> res$alternative
[1] "two.sided"
```

Test di Fisher con due campioni

- **Package:** `stats`
- **Sintassi:** `var.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n_x

  `y` vettore numerico di dimensione n_y

  `ratio` il valore di $\frac{\sigma_x^2}{\sigma_y^2} \Big| H_0$

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica F

  `parameter` gradi di libertà

  `p.value` p-value

  `conf.int` intervallo di confidenza per il rapporto tra le varianze incognite al livello $1 - \alpha$

  `estimate` rapporto tra le varianze campionarie

  `null.value` valore di $\frac{\sigma_x^2}{\sigma_y^2} \Big| H_0$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$Fval = \frac{s_x^2}{s_y^2} \frac{1}{\frac{\sigma_x^2}{\sigma_y^2} \Big| H_0}$$

  `parameter`

$$df_1 = n_x - 1 \qquad df_2 = n_y - 1$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$P(F_{df_1,df_2} \leq Fval)$	$P(F_{df_1,df_2} \geq Fval)$	$2 \min(P(F_{df_1,df_2} \leq Fval), P(F_{df_1,df_2} \geq Fval))$

  `conf.int`

$$\frac{1}{F_{1-\frac{\alpha}{2},df_1,df_2}} \frac{s_x^2}{s_y^2} \qquad \frac{1}{F_{\frac{\alpha}{2},df_1,df_2}} \frac{s_x^2}{s_y^2}$$

  `estimate`

$$\frac{s_x^2}{s_y^2}$$

  `null.value`

$$\frac{\sigma_x^2}{\sigma_y^2} \Big| H_0$$

- **Esempio:**

```
> x
[1]  7 -4 18 17 -3 -5  1 10 11 -2 -3
> nx<-length(x)
> nx
[1] 11
> y
[1]  -1  12  -1  -3   3  -5   5   2 -11  -1  -3
> ny<-length(y)
> ny
[1] 11
> ratio<-1.3
> Fval<-sd(x)**2/sd(y)**2*(1/ratio)
> Fval
[1] 1.648524
> res<-var.test(x,y,ratio=1.3,alternative="two.sided",conf.level=0.95)
> res$statistic
       F
1.648524
> c(nx-1,ny-1)
[1] 10 10
> res$parameter
  num df denom df
      10       10
> p.value<-2*min(pf(Fval,df1=nx-1,df2=ny-1),1-pf(Fval,df1=nx-1,df2=ny-1))
> p.value
[1] 0.4430561
> res$p.value
[1] 0.4430561
> alpha<-0.05
> lower<-(1/qf(1-0.05/2,df1=nx-1,df2=ny-1))*sd(x)**2/sd(y)**2
> upper<-(1/qf(0.05/2,df1=nx-1,df2=ny-1))*sd(x)**2/sd(y)**2
> c(lower,upper)
[1] 0.5765943 7.9653858
> res$conf.int
[1] 0.5765943 7.9653858
attr(,"conf.level")
[1] 0.95
```

```
> sd(x)**2/sd(y)**2
[1] 2.143081
> res$estimate
ratio of variances
          2.143081
> ratio
[1] 1.3
> res$null.value
ratio of variances
              1.3
> res$alternative
[1] "two.sided"

> x
[1] 7.8 6.6 6.5 7.4 7.3 7.0 6.4 7.1 6.7 7.6 6.8
> nx<-length(x)
> nx
[1] 11
> y
[1] 4.5 5.4 6.1 6.1 5.4 5.0 4.1 5.5
> ny<-length(y)
> ny
[1] 8
> ratio<-1.1
> Fval<-sd(x)**2/sd(y)**2*(1/ratio)
> Fval
[1] 0.3922062
> res<-var.test(x,y,ratio=1.1,alternative="two.sided",conf.level=0.95)
> res$statistic
        F
0.3922062
> c(nx-1,ny-1)
[1] 10  7
> res$parameter
  num df denom df
      10        7
> p.value<-2*min(pf(Fval,df1=nx-1,df2=ny-1),1-pf(Fval,df1=nx-1,df2=ny-1))
> p.value
[1] 0.1744655
> res$p.value
[1] 0.1744655
> alpha<-0.05
> lower<-(1/qf(1-0.05/2,df1=nx-1,df2=ny-1))*sd(x)**2/sd(y)**2
> upper<-(1/qf(0.05/2,df1=nx-1,df2=ny-1))*sd(x)**2/sd(y)**2
> c(lower,upper)
[1] 0.09061463 1.70405999
> res$conf.int
[1] 0.09061463 1.70405999
attr(,"conf.level")
[1] 0.95
> sd(x)**2/sd(y)**2
[1] 0.4314268
> res$estimate
ratio of variances
         0.4314268
> ratio
[1] 1.1
> res$null.value
ratio of variances
              1.1
> res$alternative
[1] "two.sided"
```

6.4 Test di ipotesi su proporzioni

Test con un campione

- **Package:** `stats`
- **Sintassi:** `prop.test()`
- **Parametri:**

  `x` numero di successi

  `n` dimensione campionaria

  `p` il valore di p_0

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

  `conf.int` intervallo di confidenza per la proporzione incognita al livello $1 - \alpha$

  `estimate` proporzione calcolata sulla base del campione

  `null.value` il valore di p_0

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

  $$z^2 = \left(\frac{\frac{x}{n} - p_0}{\sqrt{\frac{p_0 \, (1-p_0)}{n}}} \right)^2$$

  `parameter`

  $$1$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$\Phi(z)$	$1 - \Phi(z)$	$P(\chi_1^2 \geq z^2)$

  `conf.int`

  $$\frac{\frac{z^2_{1-\alpha\,/\,2}}{2\,n} + \frac{x}{n} \mp z_{1-\alpha\,/\,2} \sqrt{\frac{z^2_{1-\alpha\,/\,2}}{4\,n^2} + \frac{\frac{x}{n}\left(1-\frac{x}{n}\right)}{n}}}{1 + \frac{z^2_{1-\alpha\,/\,2}}{n}}$$

  `estimate`

  $$\frac{x}{n}$$

  `null.value`

  $$p_0$$

- **Esempio:**

```
> x<-10
> n<-23
> p0<-0.45
> z<-(x/n-p0)/sqrt(p0*(1-p0)/n)
> z
[1] -0.1466954
> z**2
[1] 0.02151954
> res<-prop.test(x=10,n=23,p=0.45,alternative="two.sided",conf.level=0.95,correct=F)
> res$statistic
```

```
 X-squared
0.02151954
> res$parameter
df
 1
> p.value<-1-pchisq(z**2,df=1)
> p.value
[1] 0.8833724
> res$p.value
[1] 0.8833724
> alpha<-0.05
> zc<-qnorm(1-0.05/2)
> lower<-(zc**2/(2*n)+x/n-zc*sqrt(zc**2/(4*n**2)+x/n*(1-x/n)/n))/(1+zc**2/n)
> upper<-(zc**2/(2*n)+x/n+zc*sqrt(zc**2/(4*n**2)+x/n*(1-x/n)/n))/(1+zc**2/n)
> c(lower,upper)
[1] 0.2563464 0.6318862
> res$conf.int
[1] 0.2563464 0.6318862
attr(,"conf.level")
[1] 0.95
> x/n
[1] 0.4347826
> res$estimate
        p
0.4347826
> p0
[1] 0.45
> res$null.value
   p
0.45
> res$alternative
[1] "two.sided"

> x<-18
> n<-30
> p0<-0.55
> z<-(x/n-p0)/sqrt(p0*(1-p0)/n)
> z
[1] 0.5504819
> z**2
[1] 0.3030303
> res<-prop.test(x=18,n=30,p=0.55,alternative="two.sided",conf.level=0.95,correct=F)
> res$statistic
X-squared
0.3030303
> res$parameter
df
 1
> p.value<-1-pchisq(z**2,df=1)
> p.value
[1] 0.5819889
> res$p.value
[1] 0.5819889
> alpha<-0.05
> zc<-qnorm(1-0.05/2)
> lower<-(zc**2/(2*n)+x/n-zc*sqrt(zc**2/(4*n**2)+x/n*(1-x/n)/n))/(1+zc**2/n)
> upper<-(zc**2/(2*n)+x/n+zc*sqrt(zc**2/(4*n**2)+x/n*(1-x/n)/n))/(1+zc**2/n)
> c(lower,upper)
[1] 0.4232036 0.7540937
> res$conf.int
[1] 0.4232036 0.7540937
attr(,"conf.level")
[1] 0.95
```

```
> x/n
[1] 0.6
> res$estimate
  p
0.6
> p0
[1] 0.55
> res$null.value
   p
0.55
> res$alternative
[1] "two.sided"
```

Potenza nel Test con un campione

- **Package:** `stats`
- **Sintassi:** `power.prop.test()`
- **Parametri:**

  `n` il valore n della dimensione di ciascun campione

  `p1` valore p_1 della proporzione sotto ipotesi nulla

  `p2` il valore p_2 della proporzione sotto l'ipotesi alternativa

  `sig.level` livello di significatività α

  `power` potenza $1 - \beta$

  `alternative` può essere cambiata in `one.sided`, `two.sided` a seconda del numero di code che interessano

- **Output:**

  `p1` il valore p_1 della proporzione sotto l'ipotesi nulla

  `p2` il valore p_2 della proporzione sotto l'ipotesi alternativa

  `n` il valore n della dimensione di ciascun campione

  `sig.level` livello di significatività α

  `power` potenza $1 - \beta$

  `alternative` ipotesi alternativa

- **Formula:**

$$\xi = \sqrt{p_1 (1 - p_1) + p_2 (1 - p_2)}$$

$$\delta = \sqrt{(p_1 + p_2) (1 - (p_1 + p_2) / 2)}$$

$$\gamma = |p_1 - p_2|$$

`alternative = one.sided`

`p1`

$$p_1$$

`p2`

$$p_2$$

`n`

$$n = \left[(\xi / \gamma) \Phi^{-1}(1 - \beta) + (\delta / \gamma) \Phi^{-1}(1 - \alpha) \right]^2$$

`sig.level`

$$\alpha = 1 - \Phi \left((\gamma / \delta) \sqrt{n} - (\xi / \delta) \Phi^{-1}(1 - \beta) \right)$$

`power`

$$1 - \beta = \Phi \left((\gamma / \xi) \sqrt{n} - (\delta / \xi) \Phi^{-1}(1 - \alpha) \right)$$

`alternative = two.sided`

`p1`

$$p_1$$

`p2`

$$p_2$$

`n`

$$n = \left[(\xi / \gamma)\, \Phi^{-1}(1-\beta) + (\delta / \gamma)\, \Phi^{-1}(1-\alpha/2)\right]^2$$

`sig.level`

$$\alpha = 2\left[1 - \Phi\left((\gamma / \delta)\sqrt{n} - (\xi / \delta)\, \Phi^{-1}(1-\beta)\right)\right]$$

`power`

$$1-\beta = \Phi\left((\gamma / \xi)\sqrt{n} - (\delta / \xi)\, \Phi^{-1}(1-\alpha/2)\right)$$

- **Esempio:**

```
> n<-23
> p1<-0.23
> p2<-0.31
> power.prop.test(n,p1,p2,sig.level=NULL,power=0.9,alternative="one.sided")
> # risolve rispetto ad alpha

> p1<-0.23
> p2<-0.31
> power.prop.test(n=NULL,p1,p2,sig.level=0.05,power=0.9,alternative="one.sided")
> # risolve rispetto ad n

> n<-23
> p1<-0.23
> p2<-0.31
> power.prop.test(n,p1,p2,sig.level=0.05,power=NULL,alternative="one.sided")
> # risolve rispetto a power
```

Test con due campioni indipendenti

- **Package:** `stats`
- **Sintassi:** `prop.test()`
- **Parametri:**

  `x` rappresenta il numero di successi nel primo campione

  `y` rappresenta il numero di successi nel secondo campione

  `nx` dimensione del primo campione

  `ny` dimensione del secondo campione

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1-\alpha$

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

  `conf.int` intervallo di confidenza per la differenza tra le proporzioni incognite al livello $1-\alpha$

  `estimate` proporzioni calcolate sulla base dei campioni

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$z^2 = \left(\frac{\frac{x}{n_x} - \frac{y}{n_y}}{\sqrt{\frac{x+y}{n_x+n_y}\left(1 - \frac{x+y}{n_x+n_y}\right)\left(\frac{1}{n_x} + \frac{1}{n_y}\right)}} \right)^2$$

  `parameter`

$$1$$

  `p.value`

alternative	less	greater	two.sided
p.value	$\Phi(z)$	$1 - \Phi(z)$	$1 - P(\chi_1^2 \leq z^2)$

  `conf.int`

$$\frac{x}{n_x} - \frac{y}{n_y} \mp z_{1-\alpha\,/\,2} \sqrt{\frac{\frac{x}{n_x}\left(1 - \frac{x}{n_x}\right)}{n_x} + \frac{\frac{y}{n_y}\left(1 - \frac{y}{n_y}\right)}{n_y}}$$

  `estimate`

$$\frac{x}{n_x} \qquad \frac{y}{n_y}$$

- **Esempio:**

```
> x<-9
> nx<-23
> y<-11
> ny<-32
> z<-(x/nx-y/ny)/sqrt((x+y)/(nx+ny)*(1-(x+y)/(nx+ny))*(1/nx+1/ny))
> z**2
[1] 0.1307745
> res<-prop.test(c(x,y),c(nx,ny),alternative="two.sided",conf.level=0.95,correct=F)
> res$statistic
X-squared 0.1307745
> res$parameter
df
 1
> p.value<-1-pchisq(z**2,df=1)
> p.value
[1] 0.7176304
> res$p.value
[1] 0.7176304
> lower<-(x/nx-y/ny)-qnorm(1-0.05/2)*sqrt(x/nx*(1-x/nx)/nx+y/ny*(1-y/ny)/ny)
> upper<-(x/nx-y/ny)+qnorm(1-0.05/2)*sqrt(x/nx*(1-x/nx)/nx+y/ny*(1-y/ny)/ny)
> c(lower,upper)
[1] -0.2110231  0.3061318
> res$conf.int
[1] -0.2110231  0.3061318
attr(,"conf.level")
[1] 0.95
> c(x/nx,y/ny)
[1] 0.3913043 0.3437500
> res$estimate
   prop 1    prop 2
0.3913043 0.3437500
> res$alternative
[1] "two.sided"

> x<-4
> nx<-20
> y<-11
> ny<-24
> z<-(x/nx-y/ny)/sqrt((x+y)/(nx+ny)*(1-(x+y)/(nx+ny))*(1/nx+1/ny))
> z**2
```

```
[1] 3.240153
> res<-prop.test(c(x,y),c(nx,ny),alternative="two.sided",conf.level=0.95,correct=F)
> res$statistic
X-squared
 3.240153
> res$parameter
df
 1
> p.value<-1-pchisq(z**2,df=1)
> p.value
[1] 0.07185392
> res$p.value
[1] 0.07185392
> lower<-(x/nx-y/ny)-qnorm(1-0.05/2)*sqrt(x/nx*(1-x/nx)/nx+y/ny*(1-y/ny)/ny)
> upper<-(x/nx-y/ny)+qnorm(1-0.05/2)*sqrt(x/nx*(1-x/nx)/nx+y/ny*(1-y/ny)/ny)
> c(lower,upper)
[1] -0.523793280  0.007126613
> res$conf.int
[1] -0.523793280  0.007126613
attr(,"conf.level")
[1] 0.95
> c(x/nx,y/ny)
[1] 0.2000000 0.4583333
> res$estimate
   prop 1    prop 2
0.2000000 0.4583333
> res$alternative
[1] "two.sided"
```

Test con k campioni indipendenti

- **Package:** `stats`
- **Sintassi:** `prop.test()`
- **Parametri:**

  `x` numero di successi nei k campioni

  `n` dimensione dei k campioni

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

  `estimate` proporzioni calcolate sulla base dei k campioni

- **Formula:**

  `statistic`

$$c = \sum_{i=1}^{k} \left(\frac{\frac{x_i}{n_i} - \bar{p}}{\sqrt{\bar{p}\,(1-\bar{p})\,/\,n_i}} \right)^2$$

$$\text{dove} \quad \bar{p} = \frac{\sum_{j=1}^{k} x_j}{\sum_{j=1}^{k} n_j}$$

  `parameter`

$$k - 1$$

  `p.value`

$$P(\chi^2_{k-1} \geq c)$$

`estimate`

$$\frac{x_i}{n_i} \quad \forall\, i = 1, 2, \ldots, k$$

- **Esempio:**

```
> k<-3
> x<-c(1,2,3)
> n<-c(3,5,8)
> prop.test(x,n,correct=F)
```

6.5 Test di ipotesi sull'omogeneità delle varianze

Test di Bartlett

- **Package:** `stats`
- **Sintassi:** `bartlett.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `g` fattore a k livelli di dimensione n

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

- **Formula:**

  `statistic`

$$c = \frac{(n-k)\,\log\left(s_P^2\right) - \sum_{j=1}^{k}\left(n_j - 1\right)\,\log\left(s_j^2\right)}{1 + \frac{1}{3\,(k-1)}\left(\sum_{j=1}^{k}\,\frac{1}{n_j - 1} - \frac{1}{n-k}\right)}$$

$$\text{dove} \quad s_P^2 \,=\, \frac{\sum_{j=1}^{k}\left(n_j - 1\right) s_j^2}{n-k}$$

  `parameter`

$$k - 1$$

  `p.value`

$$P(\chi_{k-1}^2 \,\geq\, c)$$

- **Esempio:**

```
> x
[1]  1.0  4.0 10.0  2.1  3.5  5.6  8.4 12.0 16.5 22.0  1.2  3.4
> g
[1] 1 1 1 2 2 2 3 3 3 4 4 4
Levels: 1 2 3 4
> n<-length(g)
> n
[1] 12
> k<-nlevels(g)
> k
[1] 4
> s2<-tapply(x,g,var)
> s2
         a          b          c          d
 21.000000   3.103333  16.470000 130.573333
> enne<-tapply(x,g,length)
```

```
> enne
a b c d
3 3 3 3
> Sp2<-sum((enne-1)*s2/(n-k))
> Sp2
[1] 42.78667
> c<-((n-k)*log(Sp2)-sum((enne-1)*log(s2)))/(1+1/(3*(k-1))*(sum(1/(enne-1))-1/(n-k)))
> c
[1] 5.254231
> res<-bartlett.test(x,g)
> res$statistic
Bartlett's K-squared
            5.254231
> parameter<-k-1
> parameter
[1] 3
> res$parameter
df
 3
> p.value<-1-pchisq(c,df=k-1)
> p.value
[1] 0.1541
> res$p.value
[1] 0.1541

> x
[1]  0.7 -1.6 -0.2 -1.2 -0.1  3.4  3.7  0.8  0.0  2.0  1.9  0.8
> g
[1] 1 1 1 1 1 1 1 1 2 2 2 2
Levels: 1 2
> n<-length(g)
> n
[1] 12
> k<-nlevels(g)
> k
[1] 2
> s2<-tapply(x,g,var)
> s2
        1         2
3.8069643 0.9091667
> enne<-tapply(x,g,length)
> enne
1 2
8 4
> Sp2<-sum((enne-1)*s2/(n-k))
> Sp2
[1] 2.937625
> c<-((n-k)*log(Sp2)-sum((enne-1)*log(s2)))/(1+1/(3*(k-1))*(sum(1/(enne-1))-1/(n-k)))
> c
[1] 1.514017
> res<-bartlett.test(x,g)
> res$statistic
Bartlett's K-squared
            1.514017
> parameter<-k-1
> parameter
[1] 1
> res$parameter
df
 1
> p.value<-1-pchisq(c,df=k-1)
> p.value
[1] 0.2185271
```

```
> res$p.value
[1] 0.2185271
```

Capitolo 7

Analisi della varianza (Anova)

7.1 Simbologia

- numero di livelli dei fattori di colonna e di riga:

Anova	f (colonna)	g (riga)
ad un fattore	k	/
a due fattori senza interazione	k	h
a due fattori con interazione	k	h

- dimensione campionaria di colonna, di riga e di cella:

Anova	j-esima colonna	i-esima riga	ij-esima cella
ad un fattore	n_j	/	/
a due fattori senza interazione	hl	kl	l
a due fattori con interazione	hl	kl	l

- medie campionarie di colonna, di riga e di cella:

Anova	j-esima colonna	i-esima riga	ij-esima cella
ad un fattore	$\bar{y}_j$	/	/
a due fattori senza interazione	$\bar{y}_{\cdot j\cdot}$	$\bar{y}_{i\cdot\cdot}$	$\bar{y}_{ij\cdot}$
a due fattori con interazione	$\bar{y}_{\cdot j\cdot}$	$\bar{y}_{i\cdot\cdot}$	$\bar{y}_{ij\cdot}$

- media campionaria generale: $\bar{y}$

7.2 Comandi utili in analisi della varianza

factor()

- **Package:** `base`
- **Parametri:**

  `x` vettore numerico o alfanumerico

  `levels` etichette di livello

- **Significato:** crea un fattore
- **Esempio:**

```
> sesso<-c(rep("U",4),rep("D",4))
> sesso
[1] "U" "U" "U" "U" "D" "D" "D" "D"
> sesso<-factor(sesso,levels=c("U","D"))
> sesso
[1] U U U U D D D D
Levels: U D
> sesso<-factor(sesso,levels=c("D","U"))
```

```
> sesso
[1] U U U U D D D D
Levels: D U
> sesso<-c(rep(1,4),rep(2,4))
> sesso
[1] 1 1 1 1 2 2 2 2
> sesso<-factor(sesso)
> sesso
[1] 1 1 1 1 2 2 2 2
Levels: 1 2
> levels(sesso)<-c("U","D")
> sesso
[1] U U U U D D D D
Levels: U D
> levels(sesso)<-c("D","U")
> sesso
[1] D D D D U U U U
Levels: D U
> fattore<-factor(scan(what="character"))
1: A
2: B
3: C
4: B
5: A
6: C
7: C
8: A
9:
Read 8 items
> fattore
[1] A B C B A C C A
Levels: A B C
```

as.factor()

- **Package:** base
- **Parametri:**

  x vettore alfanumerico di dimensione n

- **Significato:** creazione di un fattore
- **Esempio:**

```
> x<-c("a","b","b","c","a","c","b","b","c","a","c","a")
> x
[1] "a" "b" "b" "c" "a" "c" "b" "b" "c" "a" "c" "a"
> x<-as.factor(x)
> x
[1] a b b c a c b b c a c a
Levels: a b c
> x<-c(1,2,3,2,3,1,3,2)
> x
[1] 1 2 3 2 3 1 3 2
> x<-as.factor(x)
> x
[1] 1 2 3 2 3 1 3 2
Levels: 1 2 3
```

relevel()

- **Package:** `base`
- **Parametri:**

  `x` fattore a k livelli

  `ref` livello di riferimento

- **Significato:** ricodificazione dei livelli di un fattore
- **Esempio:**

```
> x
[1] a b c a b b c c a b
Levels: a b c
> cbind(x)
      f
 [1,] 1
 [2,] 2
 [3,] 3
 [4,] 1
 [5,] 2
 [6,] 2
 [7,] 3
 [8,] 3
 [9,] 1
[10,] 2
> x<-relevel(x,ref="b")
> x
[1] a b c a b b c c a b
Levels: b a c
> cbind(x)
       x
 [1,]  2
 [2,]  1
 [3,]  3
 [4,]  2
 [5,]  1
 [6,]  1
 [7,]  3
 [8,]  3
 [9,]  2
[10,]  1
> x<-relevel(x,ref="c")
> x
[1] a b c a b b c c a b
Levels: c b a
> cbind(x)
      x
 [1,] 3
 [2,] 2
 [3,] 1
 [4,] 3
 [5,] 2
 [6,] 2
 [7,] 1
 [8,] 1
 [9,] 3
[10,] 2
```

by()

- **Package:** `base`
- **Parametri:**

  `data` vettore numerico y di dimensione n

  `INDICES` fattore f a k livelli

  `FUN` funzione

- **Significato:** applica `FUN` ad ogni vettore numerico per livello del fattore
- **Esempio:**

```
> y
[1] 1.2 2.3 5.6 3.5 2.5 3.8 6.8 5.7 3.7 6.4
> f
[1] a b c a b b c c a b
Levels: a b c
> g
[1] alto medio basso alto medio basso medio alto alto basso
Levels: alto basso medio

> by(data=y,INDICES=f,FUN=mean)
INDICES: a
[1] 2.8
------------------------------------------------------------
INDICES: b
[1] 3.75
------------------------------------------------------------
INDICES: c
[1] 6.033333

> by(data=y,INDICES=list(f,g),FUN=mean)
: a
: alto
[1] 2.8
------------------------------------------------------------
: b
: alto
[1] NA
------------------------------------------------------------
: c
: alto
[1] 5.7
------------------------------------------------------------
: a
: basso
[1] NA
------------------------------------------------------------
: b
: basso
[1] 5.1
------------------------------------------------------------
: c
: basso
[1] 5.6
------------------------------------------------------------
: a
: medio
[1] NA
------------------------------------------------------------
: b
: medio
```

```
[1] 2.4
------------------------------------------------------------
: c
: medio
[1] 6.8
```

tapply()

- **Package:** `base`
- **Parametri:**

  `X` vettore numerico x di dimensione n

  `INDEX` fattore f a k livelli

  `FUN` funzione

- **Significato:** applica la funzione `FUN` ad ogni gruppo di elementi di x definito dai livelli di f
- **Esempio:**

```
> x
[1] 1.2 2.3 5.6 3.5 2.5 3.8 6.8 5.7 3.7 6.4
> f
[1] a b c a b b c c a b
Levels: a b c
> g
[1] alto medio basso alto medio basso medio alto alto basso
Levels: alto basso medio
> tapply(X=x,INDEX=f,FUN=mean)
       a        b        c
2.800000 3.750000 6.033333
> tapply(X=x,INDEX=list(f,g),FUN=mean)
  alto basso medio
a  2.8    NA    NA
b   NA   5.1   2.4
c  5.7   5.6   6.8
```

gl()

- **Package:** `base`
- **Parametri:**

  `n` numero dei livelli

  `k` numero delle replicazioni

  `length` dimensione del fattore risultato

  `labels` nomi dei livelli

- **Significato:** crea un fattore
- **Esempio:**

```
> gl(n=2,k=5,labels=c("M","F"))
[1] M  M  M  M  M  F  F  F  F  F
Levels: M  F

> gl(n=2,k=1,length=10,labels=c("A","B"))
[1] A B A B A B A B A B
Levels: A B
```

ave()

- **Package:** `stats`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `f` fattore a k livelli di dimensione n

  `FUN` funzione

- **Significato:** applica e replica la funzione FUN ad ogni gruppo di elementi di x definito dai livelli di f
- **Esempio:**

```
> x
[1] 1 2 3 4 5 6 7 8
> f
[1] a a a a b b b b
Levels: a b
> mean(x[f=="a"])
[1] 2.5
> mean(x[f=="b"])
[1] 6.5
> ave(x,f,FUN=mean)
[1] 2.5 2.5 2.5 2.5 6.5 6.5 6.5 6.5
```

```
> x
[1] 1 2 3 4 5 6 7 8
> f
[1] a a a a b b b b
Levels: a b
> sum(x[f=="a"])
[1] 10
> sum(x[f=="b"])
[1] 26
> ave(x,f,FUN=sum)
[1] 10 10 10 10 26 26 26 26
```

levels()

- **Package:** `base`
- **Parametri:**

  `f` fattore a k livelli

- **Significato:** nome dei livelli
- **Esempio:**

```
> f<-factor(c(rep(1,5),rep(2,5)))
> f
[1] 1 1 1 1 1 2 2 2 2 2
Levels: 1 2
> levels(f)
[1] "1" "2"
```

nlevels()

- **Package:** `base`
- **Parametri:**

  `f` fattore a k livelli

- **Significato:** numero di livelli
- **Esempio:**

```
> f<-factor(c(rep(1,5),rep(2,5)))
> f
[1] 1 1 1 1 1 2 2 2 2 2
Levels: 1 2
> nlevels(f)
[1] 2
```

ordered()

- **Package:** `base`
- **Parametri:**

  `x` fattore a k livelli oppure stringa di caratteri

  `levels` etichette dei livelli

- **Significato:** fattore con livelli su scala ordinale
- **Esempio:**

```
> x<-factor(c(rep(1,5),rep(2,5)))
> x
[1] 1 1 1 1 1 2 2 2 2 2
Levels: 1 2
> levels(x)<-c("U","D")
> x
[1] U U U U U D D D D D
Levels: U D
> ordered(x)
[1] U U U U U D D D D D
Levels: U < D

> ordered(x=c("a","b","c","a","b","b","c","c","a","b"),levels=c("a","b","c"))
[1] a b c a b b c c a b
Levels: a < b < c
```

as.ordered()

- **Package:** `base`
- **Parametri:**

  `x` fattore a k livelli oppure stringa di caratteri

  `levels` etichette dei livelli

- **Significato:** fattore con livelli su scala ordinale
- **Esempio:**

```
> x<-factor(c(rep(1,5),rep(2,5)))
> x
[1] 1 1 1 1 1 2 2 2 2 2
Levels: 1 2
> levels(x)<-c("U","D")
> x
[1] U U U U U D D D D D
Levels: U D
> as.ordered(x)
[1] U U U U U D D D D D
Levels: U < D

> as.ordered(x=c("a","b","c","a","b","b","c","c","a","b"),levels=c("a","b","c"))
[1] a b c a b b c c a b
Levels: a < b < c
```

letters[ ]

- **Package:** `base`
- **Significato:** lettere minuscole
- **Esempio:**

```
> x<-1:6
> letters[x]
[1] "a" "b" "c" "d" "e" "f"
> x<-c(3,5,6,26)
> letters[x]
[1] "c" "e" "f" "z"
```

LETTERS[ ]

- **Package:** `base`
- **Significato:** lettere maiuscole
- **Esempio:**

```
> x<-1:6
> LETTERS[x]
[1] "A" "B" "C" "D" "E" "F"
> x<-c(3,5,6,26)
> LETTERS[x]
[1] "C" "E" "F" "Z"
```

as.numeric()

- **Package:** `base`
- **Parametri:**

  `x` fattore a k livelli

- **Significato:** nome dei livelli
- **Esempio:**

```
> x
[1] 2 3 1 1 1 3 4 4 1 2
> x<-factor(x)
> x
[1] 2 3 1 1 1 3 4 4 1 2
Levels: 1 2 3 4
> levels(x)<-c("A","B","C","D")
> x
[1] B C A A A C D D A B
Levels: A B C D
> as.numeric(x)
[1] 2 3 1 1 1 3 4 4 1 2
```

as.integer()

- **Package:** `base`
- **Parametri:**

  `x` fattore a k livelli
- **Significato:** nome dei livelli
- **Esempio:**

```
> x
[1] 2 3 1 1 1 3 4 4 1 2
> x<-factor(x)
> x
[1] 2 3 1 1 1 3 4 4 1 2
Levels: 1 2 3 4
> levels(x)<-c("A","B","C","D")
> x
[1] B C A A A C D D A B
Levels: A B C D
> as.integer(x)
[1] 2 3 1 1 1 3 4 4 1 2
```

7.3 Modelli di analisi della varianza

Anova ad un fattore

- **Sintassi:** `anova()`
- **Parametri:**

  `y` vettore numerico di dimensione n

  `f` fattore a k livelli di dimensione n
- **Output:**

  `Df` gradi di libertà

  `Sum Sq` somma dei quadrati

  `Mean Sq` media dei quadrati

  `F value` valore empirico della statistica F

  `Pr(>F)` p-value
- **Formula:**

  `Df`

  `Sum Sq`

f	$k-1$
Residuals	$n-k$

f	$\sum_{j=1}^{k} n_j \left(\bar{y}_j - \bar{y}\right)^2$
Residuals	$\sum_{j=1}^{k} \sum_{i=1}^{n_j} \left(y_{ij} - \bar{y}_j\right)^2$

`Mean Sq`

`F value`

$$Fvalue = \frac{\left[\sum_{j=1}^{k} n_j \left(\bar{y}_j - \bar{y}\right)^2\right] / (k-1)}{\left[\sum_{j=1}^{k} \sum_{i=1}^{n_j} \left(y_{ij} - \bar{y}_j\right)^2\right] / \left(n-k\right)}$$

`Pr(>F)`

$$P(F_{k-1,\,n-k} \geq Fvalue)$$

- **Esempio:**

```
> y
[1] 1.0  4.0  10.0  2.1  3.5  5.6  8.4  12.0  16.5  22.0  1.2  3.4
> f
[1] a a a b b b c c c d d d
Levels: a b c d
> anova(lm(y~f))
```

Anova a due fattori senza interazione

- **Sintassi:** `anova()`
- **Parametri:**

  `y` vettore numerico di dimensione khl

  `f` fattore a k livelli di dimensione khl

  `g` fattore a h livelli di dimensione khl

- **Output:**

  `Df` gradi di libertà

  `Sum Sq` somma dei quadrati

  `Mean Sq` media dei quadrati

  `F value` valore empirico della statistica F

  `Pr(>F)` p-value

- **Formula:**

  `Df`

  `Sum Sq`

  `Mean Sq`

  `F value`

  `Pr(>F)`

- **Esempio:**

```
> y
[1] 1.0  4.0  10.0  2.1  3.5  5.6  8.4  12.0  6.5  2.0  1.2  3.4
> f
[1] a a a a a a b b b b b b
Levels: a b
> g
[1] B A B A B A B A B A B A
Levels: A B
> table(f,g)
```

f	$\left[\sum_{j=1}^{k} n_j \left(\bar{y}_j - \bar{y}\right)^2\right] / (k-1)$
Residuals	$\left[\sum_{j=1}^{k} \sum_{i=1}^{n_j} \left(y_{ij} - \bar{y}_j\right)^2\right] / \left(n-k\right)$

f	$k-1$
g	$h-1$
Residuals	$k\,h\,l-(k+h-1)$

```
   g
f   A B
  a 3 3
  b 3 3
> n<-length(y)
> n
[1] 12
> k<-nlevels(f)
> k
[1] 2
> h<-nlevels(g)
> h
[1] 2
> l<-3
> l
[1] 3
> anova(lm(y~f+g))
```

- **Osservazioni:** Il numero di replicazioni per cella l deve essere maggiore od uguale ad uno.

Anova a due fattori con interazione

- **Sintassi:** `anova()`
- **Parametri:**

  `y` vettore numerico di dimensione khl

  `f` fattore a k livelli di dimensione khl

  `g` fattore a h livelli di dimensione khl

- **Output:**

  `Df` gradi di libertà

  `Sum Sq` somma dei quadrati

  `Mean Sq` media dei quadrati

  `F value` valore empirico della statistica F

  `Pr(>F)` p-value

- **Formula:**

  `Df`

  `Sum Sq`

  `Mean Sq`

  `F value`

  `Pr(>F)`

- **Esempio:**

```
> y
[1] 1.0  4.0  10.0  2.1  3.5  5.6  8.4  12.0  6.5  2.0  1.2  3.4
> f
[1] a a a a a a b b b b b b
Levels: a b
> g
```

f	$hl \sum_{j=1}^{k} (\bar{y}_{\cdot j \cdot} - \bar{y})^2$
g	$kl \sum_{i=1}^{h} (\bar{y}_{i \cdot \cdot} - \bar{y})^2$
Residuals	$l \sum_{j=1}^{k} \sum_{i=1}^{h} (\bar{y}_{ij \cdot} - \bar{y}_{i \cdot \cdot} - \bar{y}_{\cdot j \cdot} + \bar{y})^2 + \sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij \cdot})^2$

f	$\left[hl \sum_{j=1}^{k} (\bar{y}_{\cdot j \cdot} - \bar{y})^2\right] / (k-1)$
g	$\left[kl \sum_{i=1}^{h} (\bar{y}_{i \cdot \cdot} - \bar{y})^2\right] / (h-1)$
Residuals	$\frac{\left[l \sum_{j=1}^{k} \sum_{i=1}^{h} (\bar{y}_{ij \cdot} - \bar{y}_{i \cdot \cdot} - \bar{y}_{\cdot j \cdot} + \bar{y})^2 + \sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij \cdot})^2\right]}{[k\,h\,l - (k+h-1)]}$

```
[1] B A B A B A B A B A B A
Levels: A B
> table(f,g)
   g
f   A B
  a 3 3
  b 3 3
> n<-length(y)
> n
[1] 12
> k<-nlevels(f)
> k
[1] 2
> h<-nlevels(g)
> h
[1] 2
> l<-3
> l
[1] 3
> anova(lm(y~f+g+f:g))
```

- **Osservazioni:** Il numero di replicazioni per cella l deve essere maggiore di uno.

$F_f value$	$\frac{\left[h\,l\,\sum_{j=1}^{k}(\bar{y}_{.j.}-\bar{y})^2\right]/(k-1)}{\frac{\left[l\,\sum_{j=1}^{k}\sum_{i=1}^{h}(\bar{y}_{ij.}-\bar{y}_{i..}-\bar{y}_{.j.}+\bar{y})^2+\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij.})^2\right]}{[k\,h\,l-(k+h-1)]}}$
$F_g value$	$\frac{\left[kl\,\sum_{i=1}^{h}(\bar{y}_{i..}-\bar{y})^2\right]/(h-1)}{\frac{\left[l\,\sum_{j=1}^{k}\sum_{i=1}^{h}(\bar{y}_{ij.}-\bar{y}_{i..}-\bar{y}_{.j.}+\bar{y})^2+\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij.})^2\right]}{[k\,h\,l-(k+h-1)]}}$

f	$P(F_{k-1,\,k\,h\,l-(k+h-1)} \geq F_f value)$
g	$P(F_{h-1,\,k\,h\,l-(k+h-1))} \geq F_g value)$

f	$k-1$
g	$h-1$
$f:g$	$(k-1)\,(h-1)$
Residuals	$k\,h\,(l-1)$

f	$hl\,\sum_{j=1}^{k}(\bar{y}_{.j.}-\bar{y})^2$
g	$kl\,\sum_{i=1}^{h}(\bar{y}_{i..}-\bar{y})^2$
$f:g$	$l\,\sum_{j=1}^{k}\sum_{i=1}^{h}(\bar{y}_{ij.}-\bar{y}_{i..}-\bar{y}_{.j.}+\bar{y})^2$
Residuals	$\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij.})^2$

f	$\left[hl\,\sum_{j=1}^{k}(\bar{y}_{.j.}-\bar{y})^2\right]/(k-1)$
g	$\left[kl\,\sum_{i=1}^{h}(\bar{y}_{i..}-\bar{y})^2\right]/(h-1)$
$f:g$	$\left[l\,\sum_{j=1}^{k}\sum_{i=1}^{h}(\bar{y}_{ij.}-\bar{y}_{i..}-\bar{y}_{.j.}+\bar{y})^2\right]/[(k-1)\,(h-1)]$
Residuals	$\left[\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij.})^2\right]/[k\,h\,(l-1)]$

$F_f value$	$\frac{\left[h\,l\,\sum_{j=1}^{k}(\bar{y}_{.j.}-\bar{y})^2\right]/(k-1)}{\left[\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij.})^2\right]/[k\,h\,(l-1)]}$
$F_g value$	$\frac{\left[kl\,\sum_{i=1}^{h}(\bar{y}_{i..}-\bar{y})^2\right]/(h-1)}{\left[\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij.})^2\right]/[k\,h\,(l-1)]}$
$F_{f:g} value$	$\frac{\left[l\,\sum_{j=1}^{k}\sum_{i=1}^{h}(\bar{y}_{ij.}-\bar{y}_{i..}-\bar{y}_{.j.}+\bar{y})^2\right]/[(k-1)\,(h-1)]}{\left[\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij.})^2\right]/[kh\,(l-1)]}$

f	$P(F_{k-1,\,k\,h\,(l-1)} \geq F_f value)$
g	$P(F_{h-1,\,k\,h\,(l-1)} \geq F_g value)$
$f:g$	$P(F_{(k-1)\,(h-1),\,k\,h\,(l-1)}) \geq F_{f:g} value)$

Capitolo 8

Confronti multipli

8.1 Simbologia

- numero di livelli dei fattori di colonna e di riga:

Anova	f (colonna)	g (riga)
ad un fattore	k	/
a due fattori senza interazione	k	h
a due fattori con interazione	k	h

- dimensione campionaria di colonna, di riga e di cella:

Anova	j-esima colonna	i-esima riga	ij-esima cella
ad un fattore	n_j	/	/
a due fattori senza interazione	hl	kl	/
a due fattori con interazione	hl	kl	l

- medie campionarie di colonna, di riga e di cella:

Anova	j-esima colonna	i-esima riga	ij-esima cella
ad un fattore	$\bar{y}_j$	/	/
a due fattori senza interazione	$\bar{y}_{\cdot j \cdot}$	$\bar{y}_{i \cdot \cdot}$	$\bar{y}_{ij \cdot}$
a due fattori con interazione	$\bar{y}_{\cdot j \cdot}$	$\bar{y}_{i \cdot \cdot}$	$\bar{y}_{ij \cdot}$

- media campionaria generale: $\bar{y}$

8.2 Metodo di Tukey

Applicazione in Anova ad un fattore

- **Sintassi:** `TukeyHSD()`
- **Parametri:**

  `y` vettore numerico di dimensione n

  `f` fattore con livelli 1, 2, ..., k

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `f` intervallo di confidenza a livello $1 - \alpha$ per il fattore `f`

- **Formula:**

  `f[,1]`

$$\bar{y}_i - \bar{y}_j \quad \forall\, i > j \;=\; 1, 2, \ldots, k$$

`f[,c(2,3)]`

$$\bar{y}_i - \bar{y}_j \mp q_{1-\alpha,\,k,\,n-k}\, s_P \sqrt{1/(2\,n_i) + 1/(2\,n_j)} \quad \forall\, i > j \,=\, 1,\, 2,\, \ldots,\, k$$

$$\text{dove} \quad s_P^2 \,=\, \sum_{j=1}^{k} \sum_{i=1}^{n_j} (y_{ij} - \bar{y}_j)^2 \,/\, (n-k)$$

- **Esempio:**

```
> y
[1] 19 24 24 27 20 24 22 21 22 29 18 17
> f
[1] 1 2 3 1 2 3 1 2 3 1 2 3
Levels: 1 2 3
> n<-length(y)
> n
[1] 12
> k<-nlevels(f)
> k
[1] 3
> alpha<-0.05
> qCRITf<-qtukey(1-alpha,k,n-k)
> qCRITf
[1] 3.948492
> TukeyHSD(aov(y~f),conf.level=1-alpha)
```

Applicazione in Anova a due fattori senza interazione

- **Sintassi:** `TukeyHSD()`
- **Parametri:**

  `y` vettore numerico di dimensione khl

  `f` fattore con livelli $1,\, 2,\, \ldots,\, k$

  `g` fattore con livelli $1,\, 2,\, \ldots,\, h$

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `f` intervallo di confidenza a livello $1 - \alpha$ per il fattore `f`

  `g` intervallo di confidenza a livello $1 - \alpha$ per il fattore `g`

- **Formula:**

`f[,1]`

$$\bar{y}_{\cdot i \cdot} - \bar{y}_{\cdot j \cdot} \quad \forall\, i > j \,=\, 1,\, 2,\, \ldots,\, k$$

`f[,c(2,3)]`

$$\bar{y}_{\cdot i \cdot} - \bar{y}_{\cdot j \cdot} \mp q_{1-\alpha,\,k,\,k\,h\,l-(k+h-1)}\, s_P \,/\, \sqrt{h\,l} \quad \forall\, i > j \,=\, 1,\, 2,\, \ldots,\, k$$

$$\text{dove} \quad s_P^2 \,=\, \frac{l \sum_{j=1}^{k} \sum_{i=1}^{h} (\bar{y}_{ij\cdot} - \bar{y}_{i\cdot\cdot} - \bar{y}_{\cdot j \cdot} + \bar{y})^2 + \sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij\cdot})^2}{k\,h\,l - (k+h-1)}$$

`g[,1]`

$$\bar{y}_{i\cdot\cdot} - \bar{y}_{j\cdot\cdot} \quad \forall\, i > j \,=\, 1,\, 2,\, \ldots,\, h$$

`g[,c(2,3)]`

$$\bar{y}_{i\cdot\cdot} - \bar{y}_{j\cdot\cdot} \mp q_{1-\alpha,\,h,\,k\,h\,l-(k+h-1)}\, s_P \,/\, \sqrt{k\,l} \quad \forall\, i > j \,=\, 1,\, 2,\, \ldots,\, h$$

$$\text{dove} \quad s_P^2 \,=\, \frac{l \sum_{j=1}^{k} \sum_{i=1}^{h} (\bar{y}_{ij\cdot} - \bar{y}_{i\cdot\cdot} - \bar{y}_{\cdot j \cdot} + \bar{y})^2 + \sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij\cdot})^2}{k\,h\,l - (k+h-1)}$$

- **Esempio:**

```
> y
[1]  1.0  4.0 10.0  2.1  3.5  5.6  8.4 12.0 16.5 22.0  1.2 3.4
> f
[1] a a a a a a b b b b b b
Levels: a b
> g
[1] A B A B A B A B A B A B
Levels: B A
> table(f,g)
   g
f   B A
  a 3 3
  b 3 3
> n<-length(y)
> n
[1] 12
> k<-nlevels(f)
> k
[1] 2
> h<-nlevels(g)
> h
[1] 2
> l<-3
> l
[1] 3
> alpha<-0.05
> qCRITf<-qtukey(1-alpha,k,k*h*l-(k+h-1))
> qCRITf
[1] 3.199173
> qCRITg<-qtukey(1-alpha,h,k*h*l-(k+h-1))
> qCRITg
[1] 3.199173
> TukeyHSD(aov(y~f+g),conf.level=0.95)
```

- **Osservazioni:** Il numero di replicazioni per cella l deve essere maggiore od uguale ad uno.

Applicazione in Anova a due fattori con interazione

- **Sintassi:** `TukeyHSD()`
- **Parametri:**

  `y` vettore numerico di dimensione khl

  `f` fattore con livelli 1, 2, ..., k

  `g` fattore con livelli 1, 2, ..., h

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `f` intervallo di confidenza a livello $1 - \alpha$ per il fattore `f`

  `g` intervallo di confidenza a livello $1 - \alpha$ per il fattore `g`

  `f:g` intervallo di confidenza a livello $1 - \alpha$ per l'interazione `f:g`

- **Formula:**

  `f[,1]`

$$\bar{y}_{\cdot i \cdot} - \bar{y}_{\cdot j \cdot} \quad \forall\, i > j \,=\, 1, 2, \ldots, k$$

  `f[,c(2,3)]`

$$\bar{y}_{\cdot i \cdot} - \bar{y}_{\cdot j \cdot} \,\mp\, q_{1-\alpha,\, k,\, k\,h\,(l-1)}\, s_P \,/\, \sqrt{h\,l} \quad \forall\, i > j \,=\, 1, 2, \ldots, k$$

$$\text{dove} \quad s_P^2 = \sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij\cdot})^2 \,/\, [k\,h\,(l-1)]$$

`g[,1]`

$$\bar{y}_{i\cdot\cdot} - \bar{y}_{j\cdot\cdot} \quad \forall\, i > j \,=\, 1,\, 2,\, \ldots,\, h$$

`g[,c(2,3)]`

$$\bar{y}_{i\cdot\cdot} - \bar{y}_{j\cdot\cdot} \mp q_{1-\alpha,\,h,\,k\,h\,(l-1)}\, s_P \,/\, \sqrt{k\,l} \quad \forall\, i > j \,=\, 1,\, 2,\, \ldots,\, h$$

$$\text{dove} \quad s_P^2 = \sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij\cdot})^2 \,/\, [k\,h\,(l-1)]$$

`f:g[,1]`

$$\bar{y}_{ij\cdot} - \bar{y}_{uw\cdot} \quad \forall\, i,\, u \,=\, 1,\, 2,\, \ldots,\, h \quad \forall\, j,\, w \,=\, 1,\, 2,\, \ldots,\, k$$

`f:g[,c(2,3)]`

$$\bar{y}_{ij\cdot} - \bar{y}_{uw\cdot} \mp q_{1-\alpha,\,k\,h,\,k\,h\,(l-1)}\, s_P \,/\, \sqrt{l} \quad \forall\, i,\, u \,=\, 1,\, 2,\, \ldots,\, h \quad \forall\, j,\, w \,=\, 1,\, 2,\, \ldots,\, k$$

$$\text{dove} \quad s_P^2 = \sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij\cdot})^2 \,/\, [k\,h\,(l-1)]$$

- **Esempio:**

```
> y
[1]  1.0  4.0 10.0  2.1  3.5  5.6  8.4 12.0 16.5 22.0  1.2 3.4
> f
[1] a a a a a a b b b b b b
Levels: a b
> g
[1] A B A B A B A B A B A B
Levels: B A
> table(f,g)
   g
f   B A
  a 3 3
  b 3 3
> n<-length(y)
> n
[1] 12
> k<-nlevels(f)
> k
[1] 2
> h<-nlevels(g)
> h
[1] 2
> l<-3
> l
[1] 3
> alpha<-0.05
> qCRITf<-qtukey(1-alpha,k,k*h*(l-1))
> qCRITf
[1] 3.261182
> qCRITg<-qtukey(1-alpha,h,k*h*(l-1))
> qCRITg
[1] 3.261182
> qCRITfg<-qtukey(1-alpha,k*h,k*h*(l-1))
> qCRITfg
[1] 4.52881
> TukeyHSD(aov(y~f+g+f:g),conf.level=0.95)
```

- **Osservazioni:** Il numero di replicazioni per cella l deve essere maggiore di uno.

8.3 Metodo di Bonferroni

Applicazione in Anova ad un fattore

- **Sintassi:** `pairwise.t.test()`
- **Parametri:**

  `y` vettore numerico di dimensione n

  `f` fattore con livelli 1, 2, ..., k livelli

- **Output:**

  `p.value` p-value

- **Formula:**

  `p.value`

$$2\binom{k}{2} P(t_{n-k} \leq -|\,t\,|) \;=\; k\,(k-1)\,P(t_{n-k} \leq -|\,t\,|)$$

$$\text{dove} \quad t \;=\; \frac{\bar{y}_i - \bar{y}_j}{s_P\,\sqrt{1\,/\,n_i + 1\,/\,n_j}} \quad \forall\, i > j \;=\; 1, 2, \ldots, k$$

$$\text{ed } s_P^2 \;=\; \sum_{j=1}^{k} \sum_{i=1}^{n_j} (y_{ij} - \bar{y}_j)^2\,/\,(n-k)$$

- **Esempio:**

```
> y
[1] 19 24 24 27 20 24 22 21 22 29 18 17
> f
[1] 1 2 3 1 2 3 1 2 3 1 2 3
Levels: 1 2 3
> n<-length(y)
> n
[1] 12
> k<-nlevels(f)
> k
[1] 3
> pairwise.t.test(y,f,p.adjust.method="bonferroni")
```

8.4 Metodo di Student

Applicazione in Anova ad un fattore

- **Sintassi:** `pairwise.t.test()`
- **Parametri:**

  `y` vettore numerico di dimensione n

  `f` fattore con livelli 1, 2, ..., k

- **Output:**

  `p.value` p-value

- **Formula:**

`p.value`

$$2\,P(t_{n-k} \leq -|\,t\,|)$$

$$\text{dove} \quad t = \frac{\bar{y}_i - \bar{y}_j}{s_P\,\sqrt{1\,/\,n_i + 1\,/\,n_j}} \quad \forall\, i > j = 1, 2, \ldots, k$$

$$\text{ed } s_P^2 = \sum_{j=1}^{k} \sum_{i=1}^{n_j} (y_{ij} - \bar{y}_j)^2\,/\,(n-k)$$

- **Esempio:**

```
> y
[1] 19 24 24 27 20 24 22 21 22 29 18 17
> f
[1] 1 2 3 1 2 3 1 2 3 1 2 3
Levels: 1 2 3
> n<-length(y)
> n
[1] 12
> k<-nlevels(f)
> k
[1] 3
> pairwise.t.test(y,f,p.adjust.method="none")
```

Capitolo 9

Test di ipotesi su correlazione ed autocorrelazione

9.1 Test di ipotesi sulla correlazione lineare

Test di Pearson

- **Package:** `stats`
- **Sintassi:** `cor.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `y` vettore numerico di dimensione n

  `alternative = less / greater / two.sided` ipotesi alternativa

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica t

  `parameter` gradi di libertà

  `p.value` p-value

  `conf.int` intervallo di confidenza a livello $1 - \alpha$ ottenuto con la trasformazione Z di *Fisher*

  `estimate` coefficiente di correlazione campionario

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$t = r_{xy} \sqrt{\frac{n-2}{1-r_{xy}^2}}$$

$$\text{dove} \quad r_{xy} = \frac{s_{xy}}{s_x \, s_y}$$

  `parameter`

$$df = n - 2$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\,P(t_{df} \leq -\lvert t \rvert)$

  `conf.int`

$$\tanh\left(\frac{1}{2}\log\left(\frac{1+r_{xy}}{1-r_{xy}}\right) \mp \frac{z_{1-\alpha\,/\,2}}{\sqrt{n-3}}\right)$$

```
    estimate
```

$$r_{xy}$$

- **Esempio:**

```
> x
[1] 1 2 2 4 3 3
> y
[1] 6 6 7 7 7 9
> n<-length(x)
> n
[1] 6
> r<-cov(x,y)/(sd(x)*sd(y))
> r
[1] 0.522233
> t<-r*sqrt((n-2)/(1-r**2))
> t
[1] 1.224745
> res<-cor.test(x,y,alternative="two.sided",conf.level=0.95,method="pearson")
> res$statistic
       t
1.224745
> parameter<-n-2
> parameter
[1] 4
> res$parameter
df
 4
> p.value<-2*pt(-abs(t),df=n-2)
> p.value
[1] 0.2878641
> res$p.value
[1] 0.2878641
> lower<-tanh(0.5*log((1+r)/(1-r))-qnorm(1-0.05/2)/sqrt(n-3))
> upper<-tanh(0.5*log((1+r)/(1-r))+qnorm(1-0.05/2)/sqrt(n-3))
> c(lower,upper)
[1] -0.5021527  0.9367690
> res$conf.int
[1] -0.5021527  0.9367690
attr(,"conf.level")
[1] 0.95
> r
[1] 0.522233
> res$estimate
     cor
0.522233
> res$alternative
[1] "two.sided"

> x
[1] 1.2 1.2 3.4 3.4 4.5 5.5 5.5 5.0 6.6 6.6 6.6
> y
[1] 1.3 1.3 1.3 4.5 5.6 6.7 6.7 6.7 8.8 8.8 9.0
> n<-length(x)
> n
[1] 11
> r<-cov(x,y)/(sd(x)*sd(y))
> r
[1] 0.9527265
> t<-r*sqrt((n-2)/(1-r**2))
> t
[1] 9.40719
> res<-cor.test(x,y,alternative="two.sided",conf.level=0.95,method="pearson")
> res$statistic
```

```
      t
9.40719
> parameter<-n-2
> parameter
[1] 9
> res$parameter
df
 9
> p.value<-2*pt(-abs(t),df=n-2)
> p.value
[1] 5.936572e-06
> res$p.value
[1] 5.936572e-06
> lower<-tanh(0.5*log((1+r)/(1-r))-qnorm(1-0.05/2)/sqrt(n-3))
> upper<-tanh(0.5*log((1+r)/(1-r))+qnorm(1-0.05/2)/sqrt(n-3))
> c(lower,upper)
[1] 0.8234897 0.9879637
> res$conf.int
[1] 0.8234897 0.9879637
attr(,"conf.level")
[1] 0.95
> r
[1] 0.9527265
> res$estimate
      cor
0.9527265
> res$alternative
[1] "two.sided"
```

Test di Kendall

- **Package:** `stats`
- **Sintassi:** `cor.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `y` vettore numerico di dimensione n

  `alternative = less / greater / two.sided` ipotesi alternativa

- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value

  `estimate` coefficiente di correlazione campionario

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$z = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \operatorname{sign}((x_j - x_i)(y_j - y_i))}{\sigma_K}$$

dove

$$\sigma_K^2 = \frac{n\,(n-1)\,(2\,n+5)}{18} + \\ - \frac{\sum_{i=1}^{g}\, t_i\,(t_i-1)\,(2\,t_i+5) + \sum_{j=1}^{h}\, u_j\,(u_j-1)\,(2\,u_j+5)}{18} + \\ + \frac{\left[\sum_{i=1}^{g}\, t_i\,(t_i-1)\,(t_i-2)\right]\,\left[\sum_{j=1}^{h}\, u_j\,(u_j-1)\,(u_j-2)\right]}{9\,n\,(n-1)\,(n-2)} + \\ + \frac{\left[\sum_{i=1}^{g}\, t_i\,(t_i-1)\right]\,\left[\sum_{j=1}^{h}\, u_j\,(u_j-1)\right]}{2\,n\,(n-1)}$$

e t, u sono i ties di x ed y rispettivamente.

`p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$\Phi(z)$	$1-\Phi(z)$	$2\,\Phi(-\,\lvert z\rvert))$

`estimate`

$$r_{xy}^K = \frac{2\,\sum_{i=1}^{n-1}\,\sum_{j=i+1}^{n}\,\mathrm{sign}((x_j-x_i)\,(y_j-y_i))}{\left(n\,(n-1)-\sum_{i=1}^{g}\,t_i\,(t_i-1)\right)^{1\,/\,2}\,\left(n\,(n-1)-\sum_{j=1}^{h}\,u_j\,(u_j-1)\right)^{1\,/\,2}}$$

- **Esempio:**

```
> x
[1] 1 2 2 4 3 3
> y
[1] 6 6 7 7 7 9
> n<-length(x)
> n
[1] 6
> matrice<-matrix(0,nrow=n-1,ncol=n,byrow=F)
> for(i in 1:(n-1))
+ for(j in (i+1):n)
+ matrice[i,j]<-sign((x[j]-x[i])*(y[j]-y[i]))
> num<-sum(matrice)
> num
[1] 7
> table(x)
x
1 2 3 4
1 2 2 1
> g<-2
> t1<-2
> t2<-2
> t<-c(t1,t2)
> t
[1] 2 2
> table(y)
y
6 7 9
2 3 1
> h<-2
> u1<-2
> u2<-3
> u<-c(u1,u2)
> u
[1] 2 3
> sigmaK<-sqrt(n*(n-1)*(2*n+5)/18-
+ (sum(t*(t-1)*(2*t+5))+sum(u*(u-1)*(2*u+5)))/18+
+ (sum(t*(t-1)*(t-2))*sum(u*(u-1)*(u-2)))/(9*n*(n-1)*(n-2))+
+ (sum(t*(t-1))*sum(u*(u-1)))/(2*n*(n-1)))
```

```
> sigmaK
[1] 4.711688
> z<-num/sigmaK
> z
[1] 1.485667
> res<-cor.test(x,y,alternative="two.sided",method="kendall",exact=F)
> res$statistic
       z
1.485667
> p.value<-2*pnorm(-abs(z))
> p.value
[1] 0.1373672
> res$p.value
[1] 0.1373672
> cor(x,y,method="kendall")
[1] 0.5853694
> res$estimate
      tau
0.5853694
> res$alternative
[1] "two.sided"

> x
 [1] 1.2 1.2 3.4 3.4 4.5 5.5 5.5 5.0 6.6 6.6 6.6
> y
 [1] 1.3 1.3 1.3 4.5 5.6 6.7 6.7 6.7 8.8 8.8 9.0
> n<-length(x)
> n
[1] 11
> matrice<-matrix(0,nrow=n-1,ncol=n,byrow=F)
> for(i in 1:(n-1))
+ for(j in (i+1):n)
+ matrice[i,j]<-sign((x[j]-x[i])*(y[j]-y[i]))
> num<-sum(matrice)
> num
[1] 45
> table(x)
x 1.2 3.4 4.5   5 5.5 6.6
  2   2   1   1   2   3
> g<-4
> t1<-2
> t2<-2
> t3<-2
> t4<-3
> t<-c(t1,t2,t3,t4)
> t
[1] 2 2 2 3
> table(y)
y 1.3 4.5 5.6 6.7 8.8   9
  3   1   1   3   2   1
> h<-3
> u1<-3
> u2<-3
> u3<-2
> u<-c(u1,u2,u3)
> u
[1] 3 3 2
> sigmaK<-sqrt(n*(n-1)*(2*n+5)/18-
+ (sum(t*(t-1)*(2*t+5))+sum(u*(u-1)*(2*u+5)))/18+
+ (sum(t*(t-1)*(t-2))*sum(u*(u-1)*(u-2)))/(9*n*(n-1)*(n-2))+
+ (sum(t*(t-1))*sum(u*(u-1)))/(2*n*(n-1)))
> sigmaK
[1] 12.27891
```

```
> z<-num/sigmaK
> z
[1] 3.664819
> res<-cor.test(x,y,alternative="two.sided",method="kendall",exact=F)
> res$statistic
       z
3.664819
> p.value<-2*pnorm(-abs(z))
> p.value
[1] 0.0002475132
> res$p.value
[1] 0.0002475132
> cor(x,y,method="kendall")
[1] 0.9278844
> res$estimate
      tau
0.9278844
> res$alternative
[1] "two.sided"
```

9.2 Test di ipotesi sulla autocorrelazione

Test di Box - Pierce

- **Package:** `stats`
- **Sintassi:** `Box.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `lag` il valore d del ritardo

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

- **Formula:**

  `statistic`

$$c = n \sum_{k=1}^{d} \hat{\rho}^2(k)$$

$$\text{dove} \quad \hat{\rho}(k) = \frac{\sum_{t=1}^{n-k} (x_t - \bar{x})(x_{t+k} - \bar{x})}{\sum_{t=1}^{n} (x_t - \bar{x})^2} \quad \forall\, k = 1, 2, \ldots, d$$

  `parameter`

$$d$$

  `p.value`

$$P(\chi_d^2 \geq c)$$

- **Esempio:**

```
> x
 [1] 1 2 7 3 5 2 0 1 4 5
> n<-length(x)
> n
[1] 10
> d<-4
> Box.test(x,lag=d,type="Box-Pierce")
```

Test di Ljung - Box

- **Package:** `stats`
- **Sintassi:** `Box.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `lag` il valore d del ritardo

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

- **Formula:**

  `statistic`

$$c = n(n+2) \sum_{k=1}^{d} \frac{1}{n-k} \hat{\rho}^2(k)$$

$$\text{dove} \quad \hat{\rho}(k) = \frac{\sum_{t=1}^{n-k} (x_t - \bar{x})(x_{t+k} - \bar{x})}{\sum_{t=1}^{n} (x_t - \bar{x})^2} \quad \forall\, k = 1, 2, \ldots, d$$

  `parameter`

$$d$$

  `p.value`

$$P(\chi^2_d \geq c)$$

- **Esempio:**

```
> x
 [1] 1 2 7 3 5 2 0 1 4 5
> n<-length(x)
> n
[1] 10
> d<-4
> Box.test(x,lag=d,type="Ljung-Box")
```

Capitolo 10

Test di ipotesi non parametrici

10.1 Simbologia

- dimensione del campione j-esimo: $n_j \quad \forall\, j = 1, 2, \ldots, k$
- media aritmetica del campione j-esimo:
  $\bar{x}_j = \frac{1}{n_j} \sum_{i=1}^{n_j} x_{ij} \quad \forall\, j = 1, 2, \ldots, k$
- varianza nel campione j-esimo:
  $s_j^2 = \frac{1}{n_j - 1} \sum_{i=1}^{n_j} (x_{ij} - \bar{x}_j)^2 \quad \forall\, j = 1, 2, \ldots, k$
- varianza *pooled*: $s_P^2 = \sum_{j=1}^{k} (n_j - 1)\, s_j^2 \,/\, (n - k)$
- somma dei ranghi nel campione j-esimo:
  $R_j \quad \forall\, j = 1, 2, \ldots, k$
- ties nel campione:
  $t_j \quad \forall\, j = 1, 2, \ldots, h$

10.2 Test di ipotesi sulla mediana con uno o due campioni

Test esatto Wilcoxon signed rank

- **Package:** `stats`
- **Sintassi:** `wilcox.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `mu` il valore di $Q_{0.5}(x)_{|\, H_0}$

  `alternative = less / greater / two.sided` ipotesi alternativa

- **Output:**

  `statistic` valore empirico della statistica V

  `p.value` p-value

  `null.value` il valore di $Q_{0.5}(x)_{|\, H_0}$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

  $$v$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$P(V \leq v)$	$P(V \geq v)$	$2 \min\left(P(V \leq v), P(V \geq v)\right)$

```
    null.value
```

$$Q_{0.5}(x)_{|\,H_0}$$

- **Esempio:**

```
> x
[1] -0.1 -0.2  0.7  0.8 -1.2 -1.6  2.0  3.4  3.7
> n<-length(x)
> n
[1] 9
> mu<-3.3
> x-mu
[1] -3.4 -3.5 -2.6 -2.5 -4.5 -4.9 -1.3  0.1  0.4
> abs(x-mu)
[1] 3.4 3.5 2.6 2.5 4.5 4.9 1.3 0.1 0.4
> # Il vettore abs(x-mu) non deve contenere valori duplicati o nulli
> xx<-rank(abs(x-mu))*sign(x-mu)
> xx
[1] -6 -7 -5 -4 -8 -9 -3  1  2
> v<-sum(xx[xx>0])
> v
[1] 3
> wilcox.test(x,mu=3.3,alternative="less",exact=T)$statistic
V
3
> p.value.less<-psignrank(v,n)
> p.value.less
[1] 0.009765625
> wilcox.test(x,mu=3.3,alternative="less",exact=T)$p.value
[1] 0.009765625
> p.value.greater<-1-psignrank(v-1,n)
> p.value.greater
[1] 0.9941406
> wilcox.test(x,mu=3.3,alternative="greater",exact=T)$p.value
[1] 0.9941406
> p.value.two.sided<-2*min(p.value.less,p.value.greater)
> p.value.two.sided
[1] 0.01953125
> wilcox.test(x,mu=3.3,alternative="two.sided",exact=T)$p.value
[1] 0.01953125

> x
[1] 3.8 5.6 1.8 5.0 2.4 4.2 7.3 8.6 9.1 5.2
> n<-length(x)
> n
[1] 10
> mu<-6.3
> x-mu
[1] -2.5 -0.7 -4.5 -1.3 -3.9 -2.1  1.0  2.3  2.8 -1.1
> abs(x-mu)
[1] 2.5 0.7 4.5 1.3 3.9 2.1 1.0 2.3 2.8 1.1
> # Il vettore abs(x-mu) non deve contenere valori duplicati o nulli
> xx<-rank(abs(x-mu))*sign(x-mu)
> xx
[1]  -7  -1 -10  -4  -9  -5   2   6   8  -3
> v<-sum(xx[xx>0])
> v
[1] 16
> wilcox.test(x,mu=6.3,alternative="less",exact=T)$statistic
 V
16
> p.value.less<-psignrank(v,n)
> p.value.less
[1] 0.1376953
```

```
> wilcox.test(x,mu=6.3,alternative="less",exact=T)$p.value
[1] 0.1376953
> p.value.greater<-1-psignrank(v-1,n)
> p.value.greater
[1] 0.883789
> wilcox.test(x,mu=6.3,alternative="greater",exact=T)$p.value
[1] 0.883789
> p.value.two.sided<-2*min(p.value.less,p.value.greater)
> p.value.two.sided
[1] 0.2753906
> wilcox.test(x,mu=6.3,alternative="two.sided",exact=T)$p.value
[1] 0.2753906
```

- **Osservazioni:** Il vettore `abs(x-mu)` non deve contenere valori duplicati o nulli.

Test asintotico Wilcoxon signed rank

- **Package:** `stats`
- **Sintassi:** `wilcox.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `mu` il valore di $Q_{0.5}(x)_{\mid H_0}$

  `alternative = less / greater / two.sided` ipotesi alternativa

  `correct = T / F` correzione di continuità di *Yates*

- **Output:**

  `statistic` valore empirico della statistica V

  `p.value` p-value

  `null.value` il valore di $Q_{0.5}(x)_{\mid H_0}$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

  $$v$$

  `p.value`

  `correct = F`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$\Phi(z)$	$1-\Phi(z)$	$2\,\Phi(-\mid z \mid)$

$$z = \frac{v - \frac{m\,(m+1)}{4}}{\left[\frac{1}{24}\left(m\,(m+1)\,(2\,m+1) - \frac{1}{2}\,\sum_{j=1}^{g}\,t_j\,(t_j^2-1)\right)\right]^{1\,/\,2}}$$

`correct = T`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$\Phi(z)$	$1-\Phi(z)$	$2\,\Phi(-\mid z \mid)$

$$z = \frac{v - \frac{m\,(m+1)}{4} + 0.5}{\left[\frac{1}{24}\left(m\,(m+1)\,(2\,m+1) - \frac{1}{2}\,\sum_{j=1}^{g}\,t_j\,(t_j^2-1)\right)\right]^{1\,/\,2}}$$

```
    null.value
```

$$Q_{0.5}(x)_{|\,H_0}$$

- **Esempio:**

```
> x
[1] 4 3 4 5 2 3 4 5 4 4 5 5 4 5 4 4 3 4 2 4 5 5 4 4
> n<-length(x)
> n
[1] 24
> mu<-4
> xx<-(x-mu)[(x-mu)!=0]
> xx
[1] -1  1 -2 -1  1  1  1  1 -1 -2  1  1
> m<-length(xx)
> m
[1] 12
> xx<-rank(abs(xx))*sign(xx)
> xx
[1]  -5.5   5.5 -11.5  -5.5   5.5   5.5   5.5   5.5  -5.5 -11.5
[11]  5.5   5.5
> v<-sum(xx[xx>0])
> v
[1] 38.5
> wilcox.test(x,mu=4,alternative="less",correct=F,exact=F)$statistic
   V
38.5
> table(rank(abs(xx)))

 5.5 11.5
  10    2
> g<-2
> t1<-10
> t2<-2
> t<-c(t1,t2)
> num<-v-m*(m+1)/4
> den<-sqrt((m*(m+1)*(2*m+1)-0.5*sum(t*(t**2-1)))/24)
> z<-num/den
> p.value<-pnorm(z)
> p.value
[1] 0.4832509
> wilcox.test(x,mu=4,alternative="less",correct=F,exact=F)$p.value
[1] 0.4832509

> x
 [1] 4 3 4 5 2 3 4 5 4 4 5 5 4 5 4 4 3 4 2 4 5 5 4 4
> n<-length(x)
> n
[1] 24
> mu<-3
> xx<-(x-mu)[(x-mu)!=0]
> xx
 [1]  1  1  2 -1  1  2  1  1  2  2  1  2  1  1  1 -1  1  2  2  1  1
> m<-length(xx)
> m
[1] 21
> xx<-rank(abs(xx))*sign(xx)
> xx
 [1]  7.5  7.5 18.0 -7.5  7.5 18.0  7.5  7.5 18.0 18.0  7.5 18.0  7.5
[14]  7.5  7.5 -7.5  7.5 18.0 18.0  7.5  7.5
> v<-sum(xx[xx>0])
> v
[1] 216
> wilcox.test(x,mu=3,alternative="less",correct=T,exact=F)$statistic
```

```
  V
216
> table(rank(abs(xx)))

7.5  18
 14   7
> g<-2
> t1<-14
> t2<-7
> t<-c(t1,t2)
> num<-v-m*(m+1)/4+0.5
> den<-sqrt((m*(m+1)*(2*m+1)-0.5*sum(t*(t**2-1)))/24)
> z<-num/den
> p.value<-pnorm(z)
> p.value
[1] 0.999871
> wilcox.test(x,mu=3,alternative="less",correct=T,exact=F)$p.value
[1] 0.999871
```

Test esatto di Mann - Whitney

- **Package:** `stats`
- **Sintassi:** `wilcox.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n_x

  `y` vettore numerico di dimensione n_y

  `mu` il valore di $(\, Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\, H_0}$

  `alternative = less / greater / two.sided` ipotesi alternativa

- **Output:**

  `statistic` valore empirico della statistica W

  `p.value` p-value

  `null.value` il valore di $(\, Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\, H_0}$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

  $$w$$

  `p.value`

alternative	less	greater	two.sided
p.value	$P(W \leq w)$	$P(W \geq w)$	$2 \min(P(W \leq w), P(W \geq w))$

  `null.value`

  $$(\, Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\, H_0}$$

- **Esempio:**

```
> x
[1]  1.2  3.4  5.4 -5.6  7.3  2.1
> nx<-length(x)
> nx
[1] 6
> y
[1] -1.1 -0.1  0.9  1.9  2.9  3.9  4.9
> ny<-length(y)
```

```
> ny
[1] 7
> mu<--2.1
> c(x,y+mu)
[1]  1.2  3.4  5.4 -5.6 7.3  2.1 -3.2 -2.2 -1.2 -0.2 0.8 1.8 2.8
> # Il vettore c(x,y+mu) non deve contenere valori duplicati
> Rx<-sum(rank(c(x,y+mu))[1:nx])
> Rx
[1] 53
> w<-Rx-nx*(nx+1)/2
> w
[1] 32
> wilcox.test(x,y,mu=-2.1,alternative="less",exact=T)$statistic
 W
32
> p.value.less<-pwilcox(w,nx,ny)
> p.value.less
[1] 0.9493007
> wilcox.test(x,y,mu=-2.1,alternative="less",exact=T)$p.value
[1] 0.9493007
> p.value.greater<-1-pwilcox(w-1,nx,ny)
> p.value.greater
[1] 0.06876457
> wilcox.test(x,y,mu=-2.1,alternative="greater",exact=T)$p.value
[1] 0.06876457
> p.value.two.sided<-2*min(p.value.less,p.value.greater)
> p.value.two.sided
[1] 0.1375291
> wilcox.test(x,y,mu=-2.1,alternative="two.sided",exact=T)$p.value
[1] 0.1375291

> x
[1] 33.30 30.10 38.62 38.94 42.63 41.96 46.30 43.25
> nx<-length(x)
> nx
[1] 8
> y
[1] 31.62 46.33 31.82 40.21 45.72 39.80 45.60 41.25
> ny<-length(y)
> ny
[1] 8
> mu<-1.1
> c(x,y+mu)
[1]  33.30 30.10 38.62 38.94 42.63 41.96 46.30 43.25 32.72 47.43
[11] 32.92 41.31 46.82 40.90 46.70 42.35
> # Il vettore c(x,y+mu) non deve contenere valori duplicati
> Rx<-sum(rank(c(x,y+mu))[1:nx])
> Rx
[1] 61
> w<-Rx-nx*(nx+1)/2
> w
[1] 25
> wilcox.test(x,y,mu=1.1,alternative="less",exact=T)$statistic
 W
25
> p.value.less<-pwilcox(w,nx,ny)
> p.value.less
[1] 0.2526807
> wilcox.test(x,y,mu=1.1,alternative="less",exact=T)$p.value
[1] 0.2526807
> p.value.greater<-1-pwilcox(w-1,nx,ny)
> p.value.greater
[1] 0.7790987
```

```
> wilcox.test(x,y,mu=1.1,alternative="greater",exact=T)$p.value
[1] 0.7790987
> p.value.two.sided<-2*min(p.value.less,p.value.greater)
> p.value.two.sided
[1] 0.5053613
> wilcox.test(x,y,mu=1.1,alternative="two.sided",exact=T)$p.value
[1] 0.5053613
```

- **Osservazioni:** Il vettore `c(x, y+mu)` non deve contenere valori duplicati.

Test asintotico di Mann - Whitney

- **Package:** `stats`
- **Sintassi:** `wilcox.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n_x

  `y` vettore numerico di dimensione n_y

  `mu` il valore di $(\, Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\, H_0}$

  `alternative = less / greater / two.sided` ipotesi alternativa

  `correct = T / F` correzione di continuità di *Yates*

- **Output:**

  `statistic` valore empirico della statistica W

  `p.value` p-value

  `null.value` il valore di $(\, Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\, H_0}$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

  $$w$$

  `p.value`

  `correct = F`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\,\lvert\, z\,\rvert)$

$$z \;=\; \frac{w - \frac{n_x\, n_y}{2}}{\left[\frac{n_x\, n_y}{12}\left(n_x + n_y + 1 - \frac{\sum_{j=1}^{g} t_j\,(t_j^2-1)}{(n_x+n_y)\,(n_x+n_y-1)}\right)\right]^{1\,/\,2}}$$

`correct = T`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\,\lvert\, z\,\rvert)$

$$z \;=\; \frac{w - \frac{n_x\, n_y}{2} + 0.5}{\left[\frac{n_x\, n_y}{12}\left(n_x + n_y + 1 - \frac{\sum_{j=1}^{g} t_j\,(t_j^2-1)}{(n_x+n_y)\,(n_x+n_y-1)}\right)\right]^{1\,/\,2}}$$

`null.value`

$$(\, Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\, H_0}$$

- **Esempio:**

```
> x
 [1] -1  1 -2 -1  1  1  1  1 -1 -2  1  1
> nx<-length(x)
> nx
[1] 12
> y
[1] 1 1 2 3 4 5 3 2 1
> ny<-length(y)
> ny
[1] 9
> mu<--4
> Rx<-sum(rank(c(x,y+mu))[1:nx])
> Rx
[1] 163.5
> w<-Rx-nx*(nx+1)/2
> w
[1] 85.5
> wilcox.test(x,y,mu=-4,alternative="less",correct=T,exact=F)$statistic
   W
85.5
> table(rank(c(x,y+mu)))

   2  5.5   10   13 17.5
   3    4    5    1    8
> g<-4
> t1<-3
> t2<-4
> t3<-5
> t4<-8
> t<-c(t1,t2,t3,t4)
> num<-w-nx*ny/2+0.5
> den<-sqrt(nx*ny/12*(nx+ny+1-sum(t*(t**2-1))/((nx+ny)*(nx+ny-1))))
> z<-num/den
> p.value<-pnorm(z)
> p.value
[1] 0.9910242
> wilcox.test(x,y,mu=-4,alternative="less",correct=T,exact=F)$p.value
[1] 0.9910242

> x
[1] 33.30 30.10 38.62 38.94 42.63 41.96 46.30 43.25
> nx<-length(x)
> nx
[1] 8
> y
[1] 31.62 46.33 31.82 40.21 45.72 39.80 45.60 41.25
> ny<-length(y)
> ny
[1] 8
> mu<-4
> Rx<-sum(rank(c(x,y+mu))[1:nx])
> Rx
[1] 51
> w<-Rx-nx*(nx+1)/2
> w
[1] 15
> wilcox.test(x,y,mu=4,alternative="less",correct=F,exact=F)$statistic
 W
15
> table(rank(x,y+mu))
```

```
1 2 3 4 5 6 7 8
1 1 1 1 1 1 1 1
> g<-8
> t1<-1
> t2<-1
> t3<-1
> t4<-1
> t5<-1
> t6<-1
> t7<-1
> t8<-1
> t<-c(t1,t2,t3,t4,t5,t6,t7,t8)
> num<-w-nx*ny/2
> den<-sqrt(nx*ny/12*(nx+ny+1-sum(t*(t**2-1))/((nx+ny)*(nx+ny-1))))
> z<-num/den
> p.value<-pnorm(z)
> p.value
[1] 0.03710171
> wilcox.test(x,y,mu=4,alternative="less",correct=F,exact=F)$p.value
[1] 0.03710171
```

Test esatto Wilcoxon signed rank per dati appaiati

- **Package:** `stats`
- **Sintassi:** `wilcox.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `y` vettore numerico di dimensione n

  `mu` il valore di $(\, Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\, H_0}$

  `alternative = less / greater / two.sided` ipotesi alternativa

- **Output:**

  `statistic` valore empirico della statistica V

  `p.value` p-value

  `null.value` il valore di $(\, Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\, H_0}$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

  $$v$$

  `p.value`

alternative	less	greater	two.sided
p.value	$P(V \leq v)$	$P(V \geq v)$	$2 \min(P(V \leq v), P(V \geq v))$

  `null.value`

  $$(\, Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\, H_0}$$

- **Esempio:**

```
> x
[1] -0.1 -0.2  0.7  0.8 -1.2 -1.6  2.0  3.4  3.7
> n<-length(x)
> n
[1] 9
```

```
> y
[1] 1 2 3 4 5 6 7 8 9
> mu<--4
> x-y-mu
[1]  2.9  1.8  1.7  0.8 -2.2 -3.6 -1.0 -0.6 -1.3
> abs(x-y-mu)
[1] 2.9 1.8 1.7 0.8 2.2 3.6 1.0 0.6 1.3
> # Il vettore abs(x-y-mu) non deve contenere valori duplicati o nulli
> xy<-rank(abs(x-y-mu))*sign(x-y-mu)
> xy
[1]  8  6  5  2 -7 -9 -3 -1 -4
> v<-sum(xy[xy>0])
> v
[1] 21
> wilcox.test(x,y,mu=-4,alternative="less",paired=T,exact=T)$statistic
 V
21
> p.value.less<-psignrank(v,n)
> p.value.less
[1] 0.4550781
> wilcox.test(x,y,mu=-4,alternative="less",paired=T,exact=T)$p.value
[1] 0.4550781
> p.value.greater<-1-psignrank(v-1,n)
> p.value.greater
[1] 0.5898438
> wilcox.test(x,y,mu=-4,alternative="greater",paired=T,exact=T)$p.value
[1] 0.5898438
> p.value.two.sided<-2*min(p.value.less,p.value.greater)
> p.value.two.sided
[1] 0.9101563
> wilcox.test(x,y,mu=-4,alternative="two.sided",paired=T,exact=T)$p.value
[1] 0.9101563

> x
[1] 33.30 30.10 38.62 38.94 42.63 41.96 46.30 43.25
> n<-length(x)
> n
[1] 8
> y
[1] 31.62 46.33 31.82 40.21 45.72 39.80 45.60 41.25
> mu<-1.1
> x-y-mu
[1]   0.58 -17.33   5.70  -2.37  -4.19   1.06  -0.40   0.90
> abs(x-y-mu)
[1]  0.58 17.33  5.70  2.37  4.19  1.06  0.40  0.90
> # Il vettore abs(x-y-mu) non deve contenere valori duplicati o nulli
> xy<-rank(abs(x-y-mu))*sign(x-y-mu)
> xy
[1]  2 -8  7 -5 -6  4 -1  3
> v<-sum(xy[xy>0])
> v
[1] 16
> wilcox.test(x,y,mu=1.1,alternative="less",paired=T,exact=T)$statistic
 V
16
> p.value.less<-psignrank(v,n)
> p.value.less
[1] 0.421875
> wilcox.test(x,y,mu=1.1,alternative="less",paired=T,exact=T)$p.value
[1] 0.421875
> p.value.greater<-1-psignrank(v-1,n)
> p.value.greater
[1] 0.6289063
```

```
> wilcox.test(x,y,mu=1.1,alternative="greater",paired=T,exact=T)$p.value
[1] 0.6289062
> p.value.two.sided<-2*min(p.value.less,p.value.greater)
> p.value.two.sided
[1] 0.84375
> wilcox.test(x,y,mu=1.1,alternative="two.sided",paired=T,exact=T)$p.value
[1] 0.84375
```

- **Osservazioni:** Il vettore `abs(x-y-mu)` non deve contenere valori duplicati o nulli.

Test asintotico Wilcoxon signed rank per dati appaiati

- **Package:** `stats`
- **Sintassi:** `wilcox.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `y` vettore numerico di dimensione n

  `mu` il valore di $(\, Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\, H_0}$

  `alternative = less / greater / two.sided` ipotesi alternativa

  `correct = T / F` correzione di continuità di *Yates*

- **Output:**

  `statistic` valore empirico della statistica V

  `p.value` p-value

  `null.value` il valore di $(\, Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\, H_0}$

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

  $$v$$

  `p.value`

  `correct = F`

alternative	less	greater	two.sided
p.value	$\Phi(z)$	$1-\Phi(z)$	$2\,\Phi(-\,|\,z\,|\,)$

$$z = \frac{v - \frac{m\,(m+1)}{4}}{\left[\frac{1}{24}\left(m\,(m+1)\,(2\,m+1) - \frac{1}{2}\,\sum_{j=1}^{g}\,t_j\,(t_j^2-1)\right)\right]^{1\,/\,2}}$$

  `correct = T`

alternative	less	greater	two.sided
p.value	$\Phi(z)$	$1-\Phi(z)$	$2\,\Phi(-\,|\,z\,|\,)$

$$z = \frac{v - \frac{m\,(m+1)}{4} + 0.5}{\left[\frac{1}{24}\left(m\,(m+1)\,(2\,m+1) - \frac{1}{2}\,\sum_{j=1}^{g}\,t_j\,(t_j^2-1)\right)\right]^{1\,/\,2}}$$

  `null.value`

$$(\, Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\, H_0}$$

- **Esempio:**

```
> x
[1] 4.0 4.0 3.0 4.0 2.0 4.0 5.0 5.0 4.0 3.3
> n<-length(x)
> n
[1] 10
> y
[1]  1  2  3  4  5  6  7  8  9 10
> mu<--2
> xy<-(x-y-mu)[(x-y-mu)!=0]
> xy
[1]  5.0  4.0  2.0  2.0 -1.0 -1.0 -3.0 -4.7
> m<-length(xy)
> m
[1] 8
> xy<-rank(abs(xy))*sign(xy)
> xy
[1]  8.0  6.0  3.5  3.5 -1.5 -1.5 -5.0 -7.0
> v<-sum(xy[xy>0])
> v
[1] 21
> wilcox.test(x,y,mu=-2,alternative="less",paired=T,exact=F,correct=T)$statistic
 V
21
> table(rank(abs(xy)))

1.5 3.5   5   6   7   8
  2   2   1   1   1   1
> g<-2
> t1<-2
> t2<-2
> t<-c(t1,t2)
> num<-v-m*(m+1)/4+0.5
> den<-sqrt(1/24*(m*(m+1)*(2*m+1)-0.5*sum(t*(t**2-1))))
> z<-num/den
> p.value<-pnorm(z)
> p.value
[1] 0.6883942
> wilcox.test(x,y,mu=-2,alternative="less",paired=T,exact=F,correct=T)$p.value
[1] 0.6883942

> x
[1] 33.30 30.10 38.62 38.94 42.63 41.96 46.30 43.25
> n<-length(x)
> n
[1] 8
> y
[1] 31.62 46.33 31.82 40.21 45.72 39.80 45.60 41.25
> mu<-2
> xy<-(x-y-mu)[(x-y-mu)!=0]
> xy
[1]  -0.32 -18.23   4.80  -3.27  -5.09   0.16  -1.30
> m<-length(xy)
> m
[1] 7
> xy<-rank(abs(xy))*sign(xy)
> xy
[1] -2 -7  5 -4 -6  1 -3
> v<-sum(xy[xy>0])
> v
[1] 6
> wilcox.test(x,y,mu=2,alternative="less",paired=T,exact=F,correct=F)$statistic
```

```
V 6
> table(rank(abs(xy)))

1 2 3 4 5 6 7
1 1 1 1 1 1 1
> g<-7
> t1<-1
> t2<-1
> t3<-1
> t4<-1
> t5<-1
> t6<-1
> t7<-1
> t<-c(t1,t2,t3,t4,t5,t6,t7)
> num<-v-m*(m+1)/4
> den<-sqrt(1/24*(m*(m+1)*(2*m+1)-0.5*sum(t*(t**2-1))))
> z<-num/den
> p.value<-pnorm(z)
> p.value
[1] 0.08814819
> wilcox.test(x,y,mu=2,alternative="less",paired=T,exact=F,correct=F)$p.value
[1] 0.08814819
```

10.3 Test di ipotesi sulla mediana con più campioni

Test di Kruskal - Wallis

- **Package:** `stats`
- **Sintassi:** `kruskal.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

  `g` fattore a k livelli di dimensione n

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

- **Formula:**

  `statistic`

  $$c = \frac{\frac{12}{n\,(n+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3\,(n+1)}{1 - \frac{\sum_{i=1}^{h} t_i\,(t_i^2-1)}{n\,(n^2-1)}}$$

  `parameter`

  $$k - 1$$

  `p.value`

  $$P(\chi^2_{k-1} \geq c)$$

- **Esempio:**

```
> x
[1] 2.1 3.0 2.1 5.3 5.3 2.1 5.6 7.5 2.1 5.3 2.1 7.5
> g
[1] a a a b b b c c c d d d
Levels: a b c d
> n<-length(x)
> n
[1] 12
> k<-nlevels(g)
```

```
> k
[1] 4
> R1<-sum(rank(x)[g=="a"])
> R2<-sum(rank(x)[g=="b"])
> R3<-sum(rank(x)[g=="c"])
> R4<-sum(rank(x)[g=="d"])
> R<-c(R1,R2,R3,R4)
> R
[1] 12.0 19.0 24.5 22.5
> table(rank(x))

   3    6    8   10 11.5
   5    1    3    1    2
> h<-3
> t1<-5
> t2<-3
> t3<-2
> t<-c(t1,t2,t3)
> t
[1] 5 3 2
> tapply(x,g,length)
a b c d
3 3 3 3
> n1<-3
> n2<-3
> n3<-3
> n4<-3
> enne<-c(n1,n2,n3,n4)
> enne
[1] 3 3 3 3
> num<-12/(n*(n+1))*sum(R**2/enne)-3*(n+1)
> den<-1-sum(t*(t**2-1))/(n*(n**2-1))
> statistic<-num/den
> statistic
[1] 2.542784
> kruskal.test(x,g)$statistic
Kruskal-Wallis chi-squared
                  2.542784
> parameter<-k-1
> parameter
[1] 3
> kruskal.test(x,g)$parameter
df
 3
> p.value<-1-pchisq(statistic,parameter)
> p.value
[1] 0.4676086
> kruskal.test(x,g)$p.value
[1] 0.4676086
```

10.4 Test di ipotesi sull'omogeneità delle varianze

Test di Levene

- **Package:** `car`
- **Sintassi:** `levene.test()`
- **Parametri:**

  `y` vettore numerico di dimensione n

  `group` fattore f a k livelli di dimensione n

- **Output:**

  `Df` gradi di libertà

  `F value` valore empirico della statistica F

  `Pr(>F)` p-value

- **Formula:**

  `Df`

f	$k-1$
Residuals	$n-k$

  `F value`

$$Fvalue = \frac{\left[\sum_{j=1}^{k}\sum_{i=1}^{n_j}(xij-\bar{x}_j)^2\right]/(k-1)}{\left[\sum_{j=1}^{k}(n_j-1)\,s_j^2\right]/(n-k)}$$

$$\text{dove} \quad x_{ij} = \left|y_{ij} - Q_{0.5}\left(\{y_{1j},\ldots,y_{n_jj}\}\right)\right| \quad \forall\, j = 1,\ldots,k \quad \forall\, i = 1,\ldots,n_j$$

  `Pr(>F)`

$$P(F_{k-1,\,n-k} \geq Fvalue)$$

- **Esempio:**

```
> y
[1]  1.0  4.0 10.0  2.1  3.5  5.6  8.4 12.0 16.5 22.0  1.2  3.4
> f
[1] a a a b b b c c c d d d
Levels: a b c d
> n<-length(f)
> n
[1] 12
> k<-nlevels(f)
> k
[1] 4
> Df<-c(k-1,n-k)
> Df
[1] 3 8
> levene.test(y,group=f)$Df
[1] 3 8
> x<-abs(y-ave(y,f,FUN="median"))
> Fvalue<-anova(lm(x~f))$F
> Fvalue
[1] 0.608269       NA
> levene.test(y,group=f)$"F value"
[1] 0.608269       NA
> p.value<-1-pf(Fvalue,k-1,n-k)
> p.value
[1] 0.6281414
> levene.test(y,group=f)$"Pr(>F)"
[1] 0.6281414
```

10.5 Anova non parametrica a due fattori senza interazione

Test di Friedman

- **Package:** `stats`
- **Sintassi:** `friedman.test()`

- **Parametri:**

  `x` matrice di dimensione $n \times k$

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

- **Formula:**

  `statistic`

  $$c = \frac{12}{n\,k\,(k+1)} \sum_{j=1}^{k} R_j^2 - 3\,n\,(k+1)$$

  `parameter`

  $$k - 1$$

  `p.value`

  $$P(\chi^2_{k-1} \geq c)$$

- **Esempio:**

```
> x
  X1 X2 X3
1  6 26 60
2 15 29 52
3  8 56 20
> n<-3
> n
[1] 3
> k<-3
> k
[1] 3
> matrice<-t(apply(x,1,rank))
> matrice
  X1 X2 X3
1  1  2  3
2  1  2  3
3  1  3  2
> colSums(matrice)
 X1  X2  X3
 3   7   8
> R1<-3
> R2<-7
> R3<-8
> R<-c(R1,R2,R3)
> R
[1] 3 7 8
> statistic<-12/(n*k*(k+1))*sum(R**2)-3*n*(k+1)
> statistic
[1] 4.666667
> friedman.test(x)$statistic
Friedman chi-squared
            4.666667
> parameter<-k-1
> parameter
[1] 2
> friedman.test(x)$parameter
df
 2
> p.value<-1-pchisq(statistic,parameter)
[1] 0.09697197
> p.value
```

```
[1] 0.09697197
> friedman.test(x)$p.value
[1] 0.09697197
```

10.6 Test di ipotesi su una proporzione

Test di Bernoulli

- **Package:** `stats`
- **Sintassi:** `binom.test()`
- **Parametri:**

  `x` numero di successi

  `n` dimensione campionaria

  `conf.level` livello di confidenza $1 - \alpha$

  `p` valore di p_0

  `alternative = less / greater / two.sided` ipotesi alternativa

- **Output:**

  `statistic` numero di successi

  `parameter` dimensione campionaria

  `p.value` p-value

  `conf.int` intervallo di confidenza per la proporzione incognita a livello $1 - \alpha$

  `estimate` proporzione campionaria

  `null.value` valore di p_0

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

  $$x$$

  `parameter`

  $$n$$

  `p.value`

  `alternative = less`

  $$\texttt{p.value} = \sum_{i=0}^{x} \binom{n}{i} p_0^i \, (1 - p_0)^{n-i}$$

  `alternative = greater`

  $$\texttt{p.value} = 1 - \sum_{i=0}^{x-1} \binom{n}{i} p_0^i \, (1 - p_0)^{n-i}$$

  `alternative = two.sided`

Caso	p.value
$x = n\,p_0$	1
$x < n\,p_0$	$F_X(x) - F_X(n-y) + 1 \quad y = \#\big(p_X(k) \le p_X(x) \quad \forall\, k = \lceil n\,p_0 \rceil, \ldots, n\big)$
$x > n\,p_0$	$F_X(y-1) - F_X(x-1) + 1 \quad y = \#\big(p_X(k) \le p_X(x) \quad \forall\, k = 0, \ldots, \lfloor n\,p_0 \rfloor\big)$

$$X \sim Binomiale(n, p_0)$$

$$p_X(x) = \binom{n}{x} p_0^x (1-p_0)^{n-x} \quad \forall x = 0, 1, \ldots, n$$

$$F_X(x) = \sum_{i=0}^{x} \binom{n}{i} p_0^i (1-p_0)^{n-i} \quad \forall x = 0, 1, \ldots, n$$

`conf.int`

$$F_U^{-1}(\alpha/2) \qquad F_H^{-1}(1-\alpha/2)$$

$$\text{dove} \quad U \sim Beta(x, n-x+1) \quad \text{e} \quad H \sim Beta(x+1, n-x)$$

`estimate`

$$\frac{x}{n}$$

`null.value`

$$p_0$$

- **Esempio:**

```
> x<-682
> n<-682+243
> p<-0.75
> binom.test(x,n,p,alternative="two.sided",conf.level=0.95)
```

10.7 Test sul ciclo di casualità

Test dei Runs

- **Package:** `tseries`
- **Sintassi:** `runs.test()`
- **Parametri:**

  `x` fattore a 2 livelli di dimensione n

  `alternative = less / greater / two.sided` ipotesi alternativa

- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$z = \frac{V - \frac{n_1+2\,n_1\,n_2+n_2}{n_1+n_2}}{\left(\frac{2\,n_1\,n_2\,(2\,n_1\,n_2-n_1-n_2)}{(n_1+n_2)^2\,(n_1+n_2-1)}\right)^{1/2}}$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$\Phi(z)$	$1-\Phi(z)$	$2\,\Phi(-\lvert z \rvert)$

- **Esempio:**

```
> x
[1] H T T H T H H H T H H T T H T H T H H T H T T H T H H T H T
Levels: H T
> n<-length(x)
> n
[1] 30
> V<-1+sum(as.numeric(x[-1]!=x[-n]))
> V
[1] 22
> n1<-length(x[x=="H"])
> n1
[1] 16
> n2<-length(x[x=="T"])
> n2
[1] 14
> media<-(n1+2*n1*n2+n2)/(n1+n2)
> media
[1] 15.93333
> varianza<-(2*n1*n2*(2*n1*n2-n1-n2))/((n1+n2)**2*(n1+n2-1))
> varianza
[1] 7.174866
> z<-(V-media)/sqrt(varianza)
> z
[1] 2.26487
> runs.test(x,alternative="less")$statistic
Standard Normal
        2.26487
> p.value<-pnorm(z)
> p.value
[1] 0.9882397
> runs.test(x,alternative="less")$p.value
[1] 0.9882397

> x
[1] a b b b a b b b a b b b a a b b a a b b a b
Levels: a b
> n<-length(x)
> n
[1] 22
> V<-1+sum(as.numeric(x[-1]!=x[-n]))
> V
[1] 12
> n1<-length(x[x=="a"])
> n1
[1] 8
> n2<-length(x[x=="b"])
> n2
[1] 14
> media<-(n1+2*n1*n2+n2)/(n1+n2)
> media
[1] 11.18182
> varianza<-(2*n1*n2*(2*n1*n2-n1-n2))/((n1+n2)**2*(n1+n2-1))
> varianza
[1] 4.451791
> z<-(V-media)/sqrt(varianza)
> z
[1] 0.3877774
> runs.test(x,alternative="two.sided")$statistic
Standard Normal
      0.3877774
> p.value<-2*pnorm(-abs(z))
```

```
> p.value
[1] 0.6981808
> runs.test(x,alternative="two.sided")$p.value
[1] 0.6981808
```

10.8 Test sulla differenza tra parametri di scala

Test di Mood

- **Package:** `stats`
- **Sintassi:** `mood.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n_x

  `y` vettore numerico di dimensione n_y

  `alternative = less / greater / two.sided` ipotesi alternativa

- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$z = \frac{T - \frac{n_x\,(n_x+n_y+1)\,(n_x+n_y-1)}{12}}{\left(\frac{n_x\,n_y\,(n_x+n_y+1)\,(n_x+n_y+2)\,(n_x+n_y-2)}{180}\right)^{1\,/\,2}}$$

  `p.value`

`alternative`	`less`	`greater`	`two.sided`
`p.value`	$\Phi(z)$	$1-\Phi(z)$	$2\,\Phi(-\,\lvert z\rvert)$

- **Esempio:**

```
> x
[1] -1  1 -2 -1  1  1  1  1 -1 -2  1  1
> y
[1] 1 2 3 4 5 6 7 8 9
> nx<-length(x)
> nx
> 12
> ny<-length(y)
> ny
> 9
> Rx<-rank(c(x,y))[1:nx]
> T<-sum((Rx-(nx+ny+1)/2)**2)
> media<-nx*(nx+ny+1)*(nx+ny-1)/12
> varianza<-nx*ny*(nx+ny+1)*(nx+ny+2)*(nx+ny-2)/180
> z<-(T-media)/sqrt(varianza)
> z
[1] -1.273865
> mood.test(x,y,alternative="less")$statistic
        Z
-1.273865
> p.value<-pnorm(z)
> p.value
[1] 0.1013557
```

```
> mood.test(x,y,alternative="less")$p.value
[1] 0.1013557

> x
[1] 1.00 4.50 6.78 9.80 7.70
> y
[1]  1.0  4.0 10.0  2.1  3.5  5.6  8.4 12.0 16.5 22.0  1.2  3.4
> nx<-length(x)
> nx
[1] 5
> ny<-length(y)
> ny
[1] 12
> Rx<-rank(c(x,y))[1:nx]
> T<-sum((Rx-(nx+ny+1)/2)**2)
> media<-nx*(nx+ny+1)*(nx+ny-1)/12
> media
[1] 120
> varianza<-nx*ny*(nx+ny+1)*(nx+ny+2)*(nx+ny-2)/180
> varianza
[1] 1710
> z<-(T-media)/sqrt(varianza)
> z
[1] -1.009621
> mood.test(x,y,alternative="two.sided")$statistic
        Z
-1.009621
> p.value<-2*pnorm(-abs(z))
> p.value
[1] 0.3126768
> mood.test(x,y,alternative="two.sided")$p.value
[1] 0.3126768
```

Capitolo 11

Tabelle di contingenza

11.1 Simbologia

- frequenze osservate: $n_{ij} \quad \forall\, i = 1, 2, \ldots, h \quad \forall\, j = 1, 2, \ldots, k$
- frequenze osservate nella m-esima tabella di contingenza 2×2:
  $n_{ijm} \quad \forall\, i, j = 1, 2 \quad \forall\, m = 1, 2, \ldots, l$
- frequenze marginali di riga: $n_{i\cdot} = \sum_{j=1}^{k} n_{ij} \quad \forall\, i = 1, 2, \ldots, h$
- frequenze marginali di riga nella m-esima tabella di contingenza 2×2:
  $n_{i\cdot m} = \sum_{j=1}^{2} n_{ijm} \quad \forall\, i = 1, 2 \quad \forall\, m = 1, 2, \ldots, l$
- frequenze marginali di colonna: $n_{\cdot j} = \sum_{i=1}^{h} n_{ij} \quad \forall\, j = 1, 2, \ldots, k$
- frequenze marginali di colonna nella m-esima tabella di contingenza 2×2:
  $n_{\cdot jm} = \sum_{i=1}^{2} n_{ijm} \quad \forall\, j = 1, 2 \quad \forall\, m = 1, 2, \ldots, l$
- frequenze attese: $\hat{n}_{ij} = \frac{n_{i\cdot}\, n_{\cdot j}}{n_{\cdot\cdot}} \quad \forall\, i = 1, 2, \ldots, h \quad \forall\, j = 1, 2, \ldots, k$
- frequenze attese nella m-esima tabella di contingenza 2×2:
  $\hat{n}_{ijm} = \frac{n_{i\cdot m}\, n_{\cdot jm}}{n_{\cdot\cdot m}} \quad \forall\, i, j = 1, 2 \quad \forall\, m = 1, 2, \ldots, l$
- totale frequenze assolute: $n_{\cdot\cdot} = \sum_{i=1}^{h} \sum_{j=1}^{k} n_{ij} = \sum_{i=1}^{h} \sum_{j=1}^{k} \hat{n}_{ij}$
- totale frequenze assolute nella m-esima tabella di contingenza 2×2:
  $n_{\cdot\cdot m} = \sum_{i=1}^{2} \sum_{j=1}^{2} n_{ijm} = \sum_{i=1}^{2} \sum_{j=1}^{2} \hat{n}_{ijm} \quad \forall\, m = 1, 2, \ldots, l$

11.2 Test di ipotesi

Test Chi - Quadrato di indipendenza

- **Sintassi:** `chisq.test()`
- **Parametri:**

  `x` matrice di dimensione 2×2 contenente frequenze assolute

  `correct = T / F` a seconda che sia applicata o meno la correzione di *Yates*

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

  `observed` frequenze osservate

  `expected` frequenze attese

  `residuals` residui di Pearson

- **Formula:**

  `statistic`

`correct = F`

$$c = \sum_{i=1}^{2}\sum_{j=1}^{2}\frac{(n_{ij}-\hat{n}_{ij})^2}{\hat{n}_{ij}} = \frac{n_{\cdot\cdot}\,(n_{11}\,n_{22}-n_{12}\,n_{21})^2}{n_{1\cdot}\,n_{2\cdot}\,n_{\cdot 1}\,n_{\cdot 2}}$$

`correct = T`

$$c = \sum_{i=1}^{2}\sum_{j=1}^{2}\frac{(|\,n_{ij}-\hat{n}_{ij}\,|-1/2)^2}{\hat{n}_{ij}} = \frac{n_{\cdot\cdot}\,(|\,n_{11}\,n_{22}-n_{12}\,n_{21}\,|-n_{\cdot\cdot}/2)^2}{n_{1\cdot}\,n_{2\cdot}\,n_{\cdot 1}\,n_{\cdot 2}}$$

`parameter`

$$1$$

`p.value`

$$P(\chi^2_1 \geq c)$$

`observed`

$$n_{ij} \quad \forall\, i,\, j = 1,\, 2$$

`expected`

$$\hat{n}_{ij} \quad \forall\, i,\, j = 1,\, 2$$

`residuals`

$$\frac{n_{ij}-\hat{n}_{ij}}{\sqrt{\hat{n}_{ij}}} \quad \forall\, i,\, j = 1,\, 2$$

- **Esempio:**

```
> x<-matrix(c(2,10,23,21),nrow=2,ncol=2,byrow=F)
> riga<-c("A","B")
> colonna<-c("A","B")
> dimnames(x)<-list(riga,colonna)
> x
   A  B
A  2 23
B 10 21
> chisq.test(x,correct=F)
```

Test di McNemar

- **Sintassi:** `mcnemar.test()`
- **Parametri:**

  `x` matrice di dimensione 2×2 contenente frequenze assolute

  `correct = T / F` a seconda che sia applicata o no la correzione di *Yates*

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

- **Formula:**

  `statistic`

`correct = F`

$$c = \frac{(n_{12}-n_{21})^2}{n_{12}+n_{21}}$$

`correct = T`

$$c = \frac{(|n_{12}-n_{21}|-1)^2}{n_{12}+n_{21}}$$

`parameter`

$$1$$

`p.value`

$$P(\chi^2_1 \geq c)$$

- **Esempio:**

```
> x<-matrix(c(2,10,23,21),nrow=2,ncol=2,byrow=F)
> riga<-c("A","B")
> colonna<-c("A","B")
> dimnames(x)<-list(riga,colonna)
> x
   A  B
A  2 23
B 10 21
> mcnemar.test(x,correct=F)
```

Test esatto di Fisher

- **Sintassi:** `fisher.test()`
- **Parametri:**

  `x` matrice di dimensione 2×2 contenente frequenze assolute

  `alt` può essere cambiata in `less`, `greater` o `two.sided` a seconda della coda che interessa

- **Output:**

  `p.value` p-value

- **Formula:**

  `p.value`

alternative	p.value
less	$\sum_{i=0}^{n_{11}} p(i)$
greater	$1 - \sum_{i=0}^{n_{11}-1} p(i)$
two.sided	$\sum_{i=0}^{n_{11}} p(i) + \sum_{p(i)\leq p(n_{11})} p(i) \quad \forall i = n_{11}+1, \ldots, \min(n_{1\cdot}, n_{\cdot 1})$

$$p(i) = \frac{\binom{\max(n_{1\cdot}, n_{\cdot 1})}{i}\binom{n_{\cdot\cdot} - \max(n_{1\cdot}, n_{\cdot 1})}{\min(n_{1\cdot}, n_{\cdot 1}) - i}}{\binom{n}{\min(n_{1\cdot}, n_{\cdot 1})}} \quad \forall i = 0, 1, \ldots, \min(n_{1\cdot}, n_{\cdot 1})$$

- **Esempio:**

```
> x<-matrix(c(2,9,5,4),nrow=2,ncol=2,byrow=F)
> riga<-c("A","B")
> colonna<-c("A","B")
> dimnames(x)<-list(riga,colonna)
> x
  A B
A 2 5
B 9 4
> n11<-2
> n1.<-2+5
> n.1<-2+9
> n..<-2+5+9+4
> n..
[1] 20
> minimo<-min(n1.,n.1)
> minimo
[1] 7
```

```
> massimo<-max(n1.,n.1)
> massimo
[1] 11
> p<-function(i) dhyper(i,massimo,n..-massimo,minimo)
> p.value.less<-0
> for(i in 0:n11){
+ p.value.less<-p.value.less+p(i)}
> p.value.less
[1] 0.1017802
> fisher.test(x,alternative="less")$p.value
[1] 0.1017802
> p.value.greater<-0
> for(i in 0:(n11-1)){
+ p.value.greater<-p.value.greater+p(i)}
> p.value.greater<-1-p.value.greater
> p.value.greater
[1] 0.9876161
> fisher.test(x,alternative="greater")$p.value
[1] 0.9876161
> p.value1<-0
> for(i in 0:n11){
+ p.value1<-p.value1+p(i)}
> p.value1
[1] 0.1017802
> p.value2<-0
> for(i in (n11+1):minimo){
+ if(p(i)<=p(n11))
+ p.value2<-p.value2+p(i)}
> p.value2
[1] 0.05789474
> p.value.two.sided<-p.value1+p.value2
> p.value.two.sided
[1] 0.1596749
> fisher.test(x,alternative="two.sided")$p.value
[1] 0.1596749
```

Test di Mantel - Haenszel

- **Sintassi:** `mantelhaen.test()`
- **Parametri:**

  `x` array di dimensione $2 \times 2 \times l$ contenente l tabelle di contingenza 2×2

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

  `estimate` stima campionaria del comune OR

  `conf.int` intervallo di confidenza a livello $1 - \alpha$

- **Formula:**

  `statistic`

$$c = \frac{\left[\sum_{m=1}^{l} (n_{11m} - \hat{n}_{11m})\right]^2}{\sum_{m=1}^{l} \hat{\sigma}^2_{n_{11m}}}$$

$$\text{dove} \quad \hat{\sigma}^2_{n_{11m}} = \frac{n_{1\cdot m}\, n_{2\cdot m}\, n_{\cdot 1m}\, n_{\cdot 2m}}{n^2_{\cdot\cdot m}\,(n_{\cdot\cdot m} - 1)} \quad \forall\, m = 1, 2, \ldots, l$$

`parameter`

$$1$$

`p.value`

$$P(\chi^2_1 \geq c)$$

`estimate`

$$\hat{\theta}_{MH} = \frac{\sum_{m=1}^{l} n_{11m}\, n_{22m} \,/\, n_{\cdot\cdot m}}{\sum_{m=1}^{l} n_{12m}\, n_{21m} \,/\, n_{\cdot\cdot m}} = \frac{\sum_{m=1}^{l} R_m}{\sum_{m=1}^{l} S_m} = \frac{R}{S}$$

`conf.int`

$$\hat{\theta}_{MH}\, e^{-z_{1-\alpha\,/\,2}\, \hat{\sigma}_{\log(\hat{\theta}_{MH})}} \quad \hat{\theta}_{MH}\, e^{z_{1-\alpha\,/\,2}\, \hat{\sigma}_{\log(\hat{\theta}_{MH})}}$$

dove

$$\hat{\sigma}^2_{\log(\hat{\theta}_{MH})} = \frac{1}{R^2} \sum_{m=1}^{l} \frac{(n_{11m} + n_{22m})\, R_m}{n_{\cdot\cdot m}} + \frac{1}{S^2} \sum_{m=1}^{l} \frac{(n_{12m} + n_{21m})\, S_m}{n_{\cdot\cdot m}} + \\ + \frac{1}{2\, R\, S} \sum_{m=1}^{l} \frac{(n_{11m} + n_{22m})\, S_m + (n_{12m} + n_{21m})\, R_m}{n_{\cdot\cdot m}}$$

- **Esempio:**

```
> x<-array(c(11,10,25,27,16,22,4,10,14,7,5,12,2,1,14,
+ 16,6,0,11,12,1,0,10, 10,1,1,4,8,4,6,2,1),dim=c(2,2,8), +
dimnames=list(Treatment=c("Drug","Control"),  +
Response=c("Success","Failure"), +
Center=c("1","2","3","4","5","6","7","8")))
> x
, , Center = 1

         Response
Treatment Success Failure
  Drug         11      25
  Control      10      27

, , Center = 2

         Response
Treatment Success Failure
  Drug         16       4
  Control      22      10

, , Center = 3

         Response
Treatment Success Failure
  Drug         14       5
  Control       7      12

, , Center = 4

         Response
Treatment Success Failure
  Drug          2      14
  Control       1      16

, , Center = 5

         Response
Treatment Success Failure
  Drug          6      11
  Control       0      12
```

```
, , Center = 6

         Response
Treatment Success Failure
  Drug          1      10
  Control       0      10

, , Center = 7

         Response
Treatment Success Failure
  Drug          1       4
  Control       1       8

, , Center = 8

         Response
Treatment Success Failure
  Drug          4       2
  Control       6       1
> mantelhaen.test(prova,conf.level=0.95,correct=F,exact=F)
```

11.3 Test di ipotesi generalizzati

Test Chi - Quadrato di indipendenza

- **Sintassi:** `chisq.test()`
- **Parametri:**

  `x` matrice di dimensione $h \times k$ contenente frequenze assolute

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

  `observed` frequenze osservate

  `expected` frequenze attese

  `residuals` residui di Pearson

- **Formula:**

  `statistic`

  $$c = \sum_{i=1}^{h}\sum_{j=1}^{k}\frac{(n_{ij}-\hat{n}_{ij})^2}{\hat{n}_{ij}} = n_{\cdot\cdot}\left(\sum_{i=1}^{h}\sum_{j=1}^{k}\frac{n_{ij}^2}{n_{i\cdot}\,n_{\cdot j}}-1\right)$$

  `parameter`

  $$df = (h-1)\,(k-1)$$

  `p.value`

  $$P(\chi^2_{df} \geq c)$$

  `observed`

  $$n_{ij} \quad \forall\, i = 1, 2, \ldots, h \quad \forall\, j = 1, 2, \ldots, k$$

  `expected`

  $$\hat{n}_{ij} \quad \forall\, i = 1, 2, \ldots, h \quad \forall\, j = 1, 2, \ldots, k$$

  `residuals`

  $$\frac{n_{ij}-\hat{n}_{ij}}{\sqrt{\hat{n}_{ij}}} \quad \forall\, i = 1, 2, \ldots, h \quad \forall\, j = 1, 2, \ldots, k$$

- **Esempio:**

```
> x<-matrix(c(2,10,23,21,11,12,43,32,30),nrow=3,ncol=3,byrow=F)
> riga<-c("A","B","C")
> colonna<-c("A","B","C")
> dimnames(x)<-list(riga,colonna)
> h<-nrow(x)
> h
[1] 3
> k<-ncol(x)
> k
[1] 3
> x
   A  B  C
A  2 21 43
B 10 11 32
C 23 12 30
> chisq.test(x)
```

Test di McNemar

- **Sintassi:** `mcnemar.test()`
- **Parametri:**

  `x` matrice di dimensione $n \times n$ contenente frequenze assolute

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

- **Formula:**

  `statistic`

  $$c = \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{(n_{ij} - n_{ji})^2}{n_{ij} + n_{ji}}$$

  `parameter`

  $$df \; = \; n\,(n-1)\,/\,2$$

  `p.value`

  $$P(\chi^2_{df} \; \geq \; c)$$

- **Esempio:**

```
> x<-matrix(c(2,10,23,21,11,12,43,32,30),nrow=3,ncol=3,byrow=F)
> riga<-c("A","B","C")
> colonna<-c("A","B","C")
> dimnames(x)<-list(riga,colonna)
> n<-nrow(x)
> n
[1] 3
> x
   A  B  C
A  2 21 43
B 10 11 32
C 23 12 30
> mcnemar.test(x)
```

11.4 Comandi utili per le tabelle di contingenza

margin.table()

- **Parametri:**

  `x` matrice di dimensione $h \times k$ contenente frequenze assolute

  `margin = 1 / 2` marginale assoluto di riga o di colonna

- **Significato:** distribuzione marginale assoluta

- **Formula:**

$$\boxed{\texttt{margin = 1}}$$

$$n_{i\cdot} \quad \forall\, i = 1, 2, \ldots, h$$

$$\boxed{\texttt{margin = 2}}$$

$$n_{\cdot j} \quad \forall\, j = 1, 2, \ldots, k$$

- **Esempio:**

```
> x<-matrix(c(1,3,0,1,3,2,2,1,2),nrow=3,ncol=3,byrow=T)
> riga<-c("a","b","c")
> colonna<-c("A","B","C")
> dimnames(x)<-list(riga,colonna)
> h<-nrow(x)
> h
[1] 3
> k<-ncol(x)
> k
[1] 3
> x
  A B C
a 1 3 0
b 1 3 2
c 2 1 2
> # distribuzione marginale assoluta di riga
> margin.table(x,margin=1)
a b c
4 6 5
> # distribuzione marginale assoluta di colonna
> margin.table(x,margin=2)
A B C
4 7 4
```

prop.table()

- **Parametri:**

  `x` matrice di dimensione $h \times k$ contenente frequenze assolute

  `margin = NULL / 1 / 2` complessiva, di riga o di colonna

- **Significato:** distribuzione relativa

- **Formula:**

$$\boxed{\texttt{margin = NULL}}$$

$$n_{ij} \,/\, n_{\cdot\cdot} \quad \forall\, i = 1, 2, \ldots, h \quad \forall\, j = 1, 2, \ldots, k$$

$$\boxed{\texttt{margin = 1}}$$

$$n_{ij} \,/\, n_{i\cdot} \quad \forall\, i = 1, 2, \ldots, h \quad \forall\, j = 1, 2, \ldots, k$$

`margin = 2`

$$n_{ij} \,/\, n_{\cdot j} \quad \forall\, i = 1, 2, \ldots, h \quad \forall\, j = 1, 2, \ldots, k$$

- **Esempio:**

```
> x<-matrix(c(1,3,0,1,3,2,2,1,2),nrow=3,ncol=3,byrow=T)
> riga<-c("a","b","c")
> colonna<-c("A","B","C")
> dimnames(x)<-list(riga,colonna)
> h<-nrow(x)
> h
[1] 3
> k<-ncol(x)
> k
[1] 3
> x
  A B C
a 1 3 0
b 1 3 2
c 2 1 2
> # distribuzione complessiva relativa
> prop.table(x,margin=NULL)
           A          B         C
a 0.06666667 0.20000000 0.0000000
b 0.06666667 0.20000000 0.1333333
c 0.13333333 0.06666667 0.1333333
> # distribuzione marginale relativa di riga
> prop.table(x,margin=1)
          A    B         C
a 0.2500000 0.75 0.0000000
b 0.1666667 0.50 0.3333333
c 0.4000000 0.20 0.4000000
> # distribuzione marginale relativa di colonna
> prop.table(x,margin=2)
     A         B   C
a 0.25 0.4285714 0.0
b 0.25 0.4285714 0.5
c 0.50 0.1428571 0.5
```

xtabs()

- **Parametri:**

  `y` vettore numerico di dimensione n

  `f` fattore a k livelli

  `g` fattore a h livelli

- **Significato:** costruzione di una tabella di contingenza a partire da un dataframe

- **Esempio:**

```
> y
[1] 1.2 2.1 1.1 2.3 5.4 4.3 3.1 2.3 4.3 5.4 5.5 5.7
> f
[1] a a a a a a b b b b b b
Levels: a b
> g
[1] B A B A B A B A B A B A
Levels: A B
```

```
> prova<-data.frame(f,g,y)
> prova
   f g   y
1  a B 1.2
2  a A 2.1
3  a B 1.1
4  a A 2.3
5  a B 5.4
6  a A 4.3
7  b B 3.1
8  b A 2.3
9  b B 4.3
10 b A 5.4
11 b B 5.5
12 b A 5.7
> xtabs(y~f+g)
   g
f    A    B
  a  8.7  7.7
  b 13.4 12.9
```

ftable()

- **Parametri:**

  `x` oggetto di tipo `table()` contenente frequenze assolute

  `row.vars` variabili di riga

  `col.vars` variabili di colonna

- **Significato:** costruzione di flat tables

- **Esempio:**

```
> Titanic
, , Age = Child, Survived = No

      Sex
Class  Male Female
  1st     0      0
  2nd     0      0
  3rd    35     17
  Crew    0      0

, , Age = Adult, Survived = No

      Sex
Class  Male Female
  1st   118      4
  2nd   154     13
  3rd   387     89
  Crew  670      3

, , Age = Child, Survived = Yes

      Sex
Class  Male Female
  1st     5      1
  2nd    11     13
  3rd    13     14
  Crew    0      0

, , Age = Adult, Survived = Yes
```

```
      Sex
Class  Male Female
  1st    57    140
  2nd    14     80
  3rd    75     76
  Crew  192     20

> ftable(x=Titanic,row.vars=c("Class","Sex","Age"),col.vars=c("Survived"))
                  Survived  No Yes
Class Sex    Age
1st   Male   Child            0   5
             Adult          118  57
      Female Child            0   1
             Adult            4 140
2nd   Male   Child            0  11
             Adult          154  14
      Female Child            0  13
             Adult           13  80
3rd   Male   Child           35  13
             Adult          387  75
      Female Child           17  14
             Adult           89  76
Crew  Male   Child            0   0
             Adult          670 192
      Female Child            0   0
             Adult            3  20

> ftable(x=Titanic,row.vars=c("Age"),col.vars=c("Sex"))
      Sex Male Female
Age
Child       64     45
Adult     1667    425
```

summary

- **Parametri:**

  `x` oggetto di tipo `table()` di dimensione $h \times k$ contenente frequenze assolute

- **Significato:** test χ^2 di indipendenza

- **Output:**

  `n.cases` totale frequenze

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

- **Formula:**

  `n.cases`

  $$n_{..}$$

  `statistic`

  $$c = \sum_{i=1}^{h}\sum_{j=1}^{k}\frac{(n_{ij}-\hat{n}_{ij})^2}{\hat{n}_{ij}} = n_{..}\left(\sum_{i=1}^{h}\sum_{j=1}^{k}\frac{n_{ij}^2}{n_{i.}\,n_{.j}}-1\right)$$

  `parameter`

  $$df = (h-1)(k-1)$$

  `p.value`

  $$P(\chi^2_{df} \geq c)$$

- **Esempio:**

```
> f
[1] a b c b a c a b b c a
Levels: a b c
> g
[1] A S A S S S A S S A A
Levels: A S
> x<-table(f,g)
> x
   g
f   A S
  a 3 1
  b 0 4
  c 2 1
> h<-nrow(x)
> h
[1] 3
> k<-ncol(x)
> k
[1] 2
> summary(x)
```

Capitolo 12

Test di adattamento

12.1 Adattamento alla distribuzione normale

Test di Kolmogorov - Smirnov

- **Sintassi:** `ks.test()`
- **Parametri:**

  `x` vettore numerico di n valori distinti

- **Significato:** test di ipotesi per $H_0 : F_0(x) = \Phi\left(\frac{x-\mu}{\sigma}\right)$ contro $H_1 : F_0(x) \neq \Phi\left(\frac{x-\mu}{\sigma}\right)$
- **Output:**

  `statistic` valore empirico della statistica D

- **Formula:**

  `statistic`

$$d = \max_{1 \leq i \leq n} \left\{ \max \left[\frac{i}{n} - F_0\left(x_{(i)}\right), F_0\left(x_{(i)}\right) - \frac{i-1}{n} \right] \right\}$$

$$\text{dove} \quad F_0\left(x_{(i)}\right) = \Phi\left(\frac{x_{(i)} - \mu}{\sigma}\right) \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> x
[1] 0.10 2.30 4.30 4.20 5.60 7.21 8.20
> n<-length(x)
> n
[1] 7
> # sono 7 valori distinti
> x<-sort(x,decreasing=F)
> x
[1] 0.10 2.30 4.20 4.30 5.60 7.21 8.20
> mu<-3.3
> sigma<-1.2
> Fo<-pnorm(x,mean=mu,sd=sigma)
> vettore1<-(1:n)/n-Fo
> vettore2<-Fo-((1:n)-1)/n
> d<-max(pmax(vettore1,vettore2))
> d
[1] 0.4876584
> ks.test(x,"pnorm",3.3,1.2)$statistic
        D
0.4876584

> x
[1] 1.1 3.4 5.6 7.8 2.3 4.5 1.2 2.2
> n<-length(x)
```

```
> n
[1] 8
> # sono 8 valori distinti
> x<-sort(x,decreasing=F)
> x
[1] 1.1 1.2 2.2 2.3 3.4 4.5 5.6 7.8
> mu<-4.1
> sigma<-2.3
> Fo<-pnorm(x,mean=mu,sd=sigma)
> vettore1<-(1:n)/n-Fo
> vettore2<-Fo-((1:n)-1)/n
> d<-max(pmax(vettore1,vettore2))
> d
[1] 0.2830715
> ks.test(x,"pnorm",4.1,2.3)$statistic
        D
0.2830715
```

Test di Jarque - Bera

- **Sintassi:** `jarque.bera.test()`
- **Parametri:**

  `x` vettore numerico di dimensione n

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

- **Formula:**

  `statistic`

$$c = \frac{n}{6}\left(\frac{m_3}{m_2^{3/2}}\right)^2 + \frac{n}{24}\left(\frac{m_4}{m_2^2} - 3\right)^2$$

$$\text{dove } m_k = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^k \quad \forall\, k = 2,\, 3,\, 4$$

  `parameter`

$$2$$

  `p.value`

$$P(\chi_2^2 \geq c)$$

- **Esempio:**

```
> x
[1] 0.10 2.30 4.30 4.20 5.60 7.21 8.20
> n<-length(x)
> n
[1] 7
> m2<-mean((x-mean(x))**2)
> m2
[1] 6.650012
> m3<-mean((x-mean(x))**3)
> m3
[1] -4.594487
> m4<-mean((x-mean(x))**4)
> m4
[1] 92.51966
```

```
> c<-(n/6)*(m3/m2**(3/2))**2+(n/24)*(m4/m2**2-3)**2
> c
[1] 0.3241426
> jarque.bera.test(x)$statistic
X-squared
0.3241426
> jarque.bera.test(x)$parameter
df
 2
> p.value<-1-pchisq(c,df=2)
> p.value
[1] 0.8503806
> jarque.bera.test(x)$p.value
X-squared
0.8503806

> x
[1] 1.1 3.4 5.6 7.8 2.3 4.5 1.2 2.2 1.1
> n<-length(x)
> n
[1] 9
> m2<-mean((x-mean(x))**2)
> m2
[1] 4.806914
> m3<-mean((x-mean(x))**3)
> m3
[1] 8.816102
> m4<-mean((x-mean(x))**4)
> m4
[1] 58.41274
> c<-(n/6)*(m3/m2**(3/2))**2+(n/24)*(m4/m2**2-3)**2
> c
[1] 1.133201
> jarque.bera.test(x)$statistic
X-squared
 1.133201
> jarque.bera.test(x)$parameter
df
 2
> p.value<-1-pchisq(c,df=2)
> p.value
[1] 0.5674513
> jarque.bera.test(x)$p.value
X-squared
0.5674513
```

- **Osservazioni:** E' necessario installare la libreria `tseries`.

Test di Cramer - von Mises

- **Sintassi:** `cvm.test()`
- **Parametri:**

  `x` vettore numerico di dimensione $n \geq 7$

- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value

- **Formula:**

`statistic`

$$W \;=\; \frac{1}{12\,n} + \sum_{i=1}^{n} \left[\Phi\Big(\frac{x_{(i)} - \bar{x}}{s_x} \Big) - \frac{2\,i\,-1}{2\,n} \right]^2$$

`p.value`

$WW \;=\; (1 + 0.5\,/\,n)\,W$

WW	< 0.0275	≥ 0.0275 AND < 0.051
p.value	$1 - e^{-13.953+775.5\,WW-12542.61\,WW^2}$	$1 - e^{-5.903+179.546\,WW-1515.29\,WW^2}$
WW	≥ 0.051 AND < 0.092	≥ 0.092
p.value	$e^{0.886-31.62\,WW+10.897\,WW^2}$	$e^{1.111-34.242\,WW+12.832\,WW^2}$

- **Esempio:**

```
> x
[1] 1.1 1.2 2.2 2.3 3.4 4.5 5.6 7.8
> n<-length(x)
> n
[1] 8
> # n>=7
> x<-sort(x,decreasing=F)
> W<-1/(12*n)+sum((pnorm((x-mean(x))/sd(x))-(2*(1:n)-1)/(2*n))**2)
> W
[1] 0.04611184
> cvm.test(x)$statistic
         W
0.04611184
> WW<-(1+0.5/n)*W
> WW
[1] 0.04899383
> # 0.0275 <= WW < 0.051
> p.value<-1-exp(-5.903+179.546*WW-1515.29*WW**2)
> p.value
[1] 0.5246239
> cvm.test(x)$p.value
[1] 0.5246239

> x
[1] 80.00 96.19 98.07 99.70 99.79 99.81 101.14 101.60 103.44 103.53
> n<-length(x)
> n
[1] 10
> # n>=7
> x<-sort(x,decreasing=F)
> W<-(1/(12*n))+sum((pnorm((x-mean(x))/sd(x))-(2*(1:n)-1)/(2*n))**2)
> W
[1] 0.2296694
> cvm.test(x)$statistic
        W
0.2296694
> WW<-(1+0.5/n)*W
> WW
[1] 0.2411529
> # WW >= 0.092
> p.value<-exp(1.111-34.242*WW+12.832*WW**2)
> p.value
[1] 0.001661032
> cvm.test(x)$p.value
[1] 0.001661032
```

- **Osservazioni:** E' necessario installare la libreria `nortest`.

Test di Anderson - Darlin

- **Sintassi:** `ad.test()`
- **Parametri:**

  `x` vettore numerico di dimensione $n \geq 7$

- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value

- **Formula:**

  `statistic`

$$A = -n - \frac{1}{n}\sum_{i=1}^{n}(2\,i\,-1)\,\left[\log\left(\Phi\Big(\frac{x_{(i)}-\bar{x}}{s_x}\Big)\right) + \log\left(1-\Phi\Big(\frac{x_{(n-i+1)}-\bar{x}}{s_x}\Big)\right)\right]$$

  `p.value`

$$AA = (1 + 0.75\,/\,n + 2.25\,/\,n^2)\,A$$

AA	< 0.2	≥ 0.2 AND < 0.34
`p.value`	$1 - e^{-13.436+101.14\,AA-223.73\,AA^2}$	$1 - e^{-8.318+42.796\,AA-59.938\,AA^2}$
AA	≥ 0.34 AND < 0.6	≥ 0.6
`p.value`	$e^{0.9177-4.279\,AA-1.38\,AA^2}$	$e^{1.2937-5.709\,AA+0.0186\,AA^2}$

- **Esempio:**

```
> x
[1] 99.70 99.79 101.14 99.32 99.27 101.29 100.30 102.40 105.20
> n<-length(x)
> n
[1] 9
> # n>=7
> x<-sort(x,decreasing=F)
> A<--n-mean((2*(1:n)-1)*(log(pnorm((x-mean(x))/sd(x))) +
+ log(1-pnorm((rev(x)-mean(x))/sd(x)))))
> A
[1] 0.5914851
> ad.test(x)$statistic
        A
0.5914851
> AA<-(1+0.75/n+2.25/n**2)*A
> AA
[1] 0.6572057
> # AA >= 0.6
> p.value<-exp(1.2937-5.709*AA+0.0186*AA**2)
> p.value
[1] 0.08627171
> ad.test(x)$p.value
[1] 0.08627171

> x
[1] 1.1 1.2 2.2 2.3 3.4 4.5 5.6 7.8
> n<-length(x)
> n
```

```
[1] 8
> # n>=7
> x<-sort(x,decreasing=F)
> A<--n-mean((2*(1:n)-1)*(log(pnorm((x-mean(x))/sd(x))) +
+ log(1-pnorm((rev(x)-mean(x))/sd(x)))))
> A
[1] 0.3073346
> ad.test(x)$statistic
       A
0.3073346
> AA<-(1+0.75/n+2.25/n**2)*A
> AA
[1] 0.346952
> # 0.34 <= AA < 0.6
> p.value<-exp(0.9177-4.279*AA-1.38*AA**2)
> p.value
[1] 0.480453
> ad.test(x)$p.value
[1] 0.480453
```

- **Osservazioni:** E' necessario installare la libreria `nortest`.

Test di Shapiro - Francia

- **Sintassi:** `sf.test()`
- **Parametri:**

  `x` vettore numerico di dimensione $5 \leq n \leq 5000$

- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value

- **Formula:**

  `statistic`

$$W = \frac{\left(\sum_{i=1}^{n} x_{(i)}\, y_i - n\,\bar{x}\,\bar{y}\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}$$

$$\text{dove} \quad y_i = \Phi^{-1}\left(\frac{i - 3/8}{n + 1/4}\right) \forall\, i = 1, 2, \ldots, n$$

  `p.value`

$$1 - \Phi(z)$$

$$\text{dove} \quad z = \frac{\log(1 - W) - [-1.2725 + 1.0521\,[\log(\log(n)) - \log(n)]]}{1.0308 - 0.26758\,[\log(\log(n)) + 2/\log(n)]}$$

- **Esempio:**

```
> x
[1] 7.7 5.6 4.3 3.2 3.1 2.2 1.2 1.0
> n<-length(x)
> n
[1] 8
> # 5<=n<=5000
> x<-sort(x,decreasing=F)
> y<-qnorm(((1:n)-3/8)/(n+1/4))
> W<-cor(x,y)**2
> W
[1] 0.9420059
> sf.test(x)$statistic
```

```
        W
0.9420059
> z<-(log(1-W)-(-1.2725+1.0521*(log(log(n))-log(n))))/
+ (1.0308-0.26758*(log(log(n))+2/log(n)))
> z
[1] -0.2724882
> p.value<-1-pnorm(z)
> p.value
[1] 0.6073767
> sf.test(x)$p.value
[1] 0.6073767

> x
[1] 1.20 3.20 4.20 2.10 0.34 3.40 9.30 9.20 9.90 10.20 11.20
> n<-length(x)
> n
[1] 11
> # 5<=n<=5000
> x<-sort(x,decreasing=F)
> y<-qnorm(((1:n)-3/8)/(n+1/4))
> W<-cor(x,y)**2
> W
[1] 0.8921455
> sf.test(x)$statistic
        W
0.8921455
> z<-(log(1-W)-(-1.2725+1.0521*(log(log(n))-log(n))))/
+ (1.0308-0.26758*(log(log(n))+2/log(n)))
> z
[1] 1.130053
> p.value<-1-pnorm(z)
> p.value
[1] 0.1292269
> sf.test(x)$p.value
[1] 0.1292269
```

- **Osservazioni:** E' necessario installare la libreria `nortest`.

Test di Lilliefors

- **Sintassi:** `lillie.test()`
- **Parametri:**

  `x` vettore numerico di dimensione $n \geq 5$

- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value

- **Formula:**

  `statistic`

$$D = \max(a, b)$$

$$\text{dove} \qquad \text{a} = \max\left\{\frac{i}{n} - \Phi\left(\frac{x_{(i)}-\bar{x}}{s_x}\right)\right\}_{i=1,\ldots,n}$$

$$\text{b} = \max\left\{\Phi\left(\frac{x_{(i)}-\bar{x}}{s_x}\right) - \frac{i-1}{n}\right\}_{i=1,\ldots,n}$$

  `p.value`

n	$n \leq 100$	$n > 100$
Kd	D	$(n \,/\, 100)^{0.49}\, D$
nd	n	100

$$pvalue = e^{-7.01256\, Kd^2\, (nd+2.78019)\, +2.99587\, Kd\, \sqrt{nd+2.78019} - 0.122119 + \frac{0.974598}{\sqrt{nd}} + \frac{1.67997}{nd}}$$

$$\boxed{pvalue \leq 0.1}$$

$$\texttt{p.value} = pvalue$$

$$\boxed{pvalue > 0.1}$$

kk	p.value
≤ 0.302	1
≤ 0.5	$2.76773 - 19.828315\, kk + 80.709644\, kk^2 - 138.55152\, kk^3 + 81.218052\, kk^4$
≤ 0.9	$-4.901232 + 40.662806\, kk - 97.490286\, kk^2 + 94.029866\, kk^3 - 32.355711\, kk^4$
≤ 1.31	$6.198765 - 19.558097\, kk + 23.186922\, kk^2 - 12.234627\, kk^3 + 2.423045\, kk^4$
> 1.31	0

$$kk = (\sqrt{n} - 0.01 + 0.85 \,/\, \sqrt{n})\, D$$

- **Esempio:**

```
> x
[1] 1.1 1.2 2.2 2.3 3.4 4.5 5.6 7.8
> n<-length(x)
> n
[1] 8
> # n>=5
> x<-sort(x,decreasing=F)
> a<-max((1:n)/n-pnorm((x-mean(x))/sd(x)))
> a
[1] 0.1983969
> b<-max(pnorm((x-mean(x))/sd(x))-((1:n)-1)/n)
> b
[1] 0.1505139
> D<-max(a,b)
> D
[1] 0.1983969
> lillie.test(x)$statistic
        D
0.1983969
> # n <= 100
> Kd<-D
> nd<-n
> pvalue<-exp(-7.01256*Kd**2*(nd+2.78019) +
+ 2.99587*Kd*sqrt(nd+2.78019)-0.122119+0.974598/sqrt(nd)+1.67997/nd)
> pvalue
[1] 0.5534262
> # pvalue > 0.1
> kk<-(sqrt(n)-0.01+0.85/sqrt(n))*D
> kk
[1] 0.6187895
> # kk <= 0.9
> p.value<--4.901232+40.662806*kk-97.490286*kk**2 +
+ 94.029866*kk**3-32.355711*kk**4
> p.value
[1] 0.4665968
> lillie.test(x)$p.value
[1] 0.4665968

> x
[1] 42.3 31.4 11.2  9.0  8.5  7.5  5.6  2.3
```

```
> n<-length(x)
> n
[1] 8
> # n>=5
> x<-sort(x,decreasing=F)
> a<-max((1:n)/n-pnorm((x-mean(x))/sd(x)))
> a
[1] 0.3479997
> b<-max(pnorm((x-mean(x))/sd(x))-((1:n)-1)/n)
> b
[1] 0.1908506
> D<-max(a,b)
> D
[1] 0.3479997
> lillie.test(x)$statistic
        D
0.3479997
> # n <= 100
> Kd<-D
> nd<-n
> pvalue<-exp(-7.01256*Kd**2*(nd+2.78019) +
+ 2.99587*Kd*sqrt(nd+2.78019)-0.122119+0.974598/sqrt(nd)+1.67997/nd)
> pvalue
[1] 0.004993897
> # pvalue <= 0.1
> p.value<-pvalue
> p.value
[1] 0.004993897
> lillie.test(x)$p.value
[1] 0.004993897
```

- **Osservazioni:** E' necessario installare la libreria `nortest`.

12.2 Adattamento ad una distribuzione nota

Test Chi - Quadrato GOF

- **Sintassi:** `chisq.test()`
- **Parametri:**

  `x` vettore di frequenze assolute a somma n di dimensione k

  `p` vettore p di probabilità a somma unitaria di dimensione k

- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

  `observed` valori osservati

  `expected` valori attesi

  `residuals` residui di *Pearson*

- **Formula:**

  `statistic`

$$c = \sum_{i=1}^{k} \frac{(n_i - \hat{n}_i)^2}{\hat{n}_i} = \sum_{i=1}^{k} \frac{n_i^2}{\hat{n}_i} - n$$

$$\text{dove} \quad \hat{n}_i = n\, p_i \quad \forall\, i = 1, 2, \ldots, k$$

`parameter`

$$k - 1$$

`p.value`

$$P(\chi^2_{k-1} \geq c)$$

`observed`

$$n_i \quad \forall\, i = 1, 2, \ldots, k$$

`expected`

$$\hat{n}_i = n\, p_i \quad \forall\, i = 1, 2, \ldots, k$$

`residuals`

$$\frac{n_i - \hat{n}_i}{\sqrt{\hat{n}_i}} \quad \forall\, i = 1, 2, \ldots, k$$

- **Esempio:**

```
> x
[1] 100 110  80  55  14
> n<-sum(x)
> n
[1] 359
> prob
[1] 0.29 0.21 0.17 0.17 0.16
> # le probabilità sommano ad 1
> k<-length(x)
> k
[1] 5
> osservati<-x
> attesi<-n*prob
> c<-sum((osservati-attesi)**2/attesi)
> c
[1] 55.3955
> chisq.test(x,p=prob)$statistic
X-squared
  55.3955
> parameter<-k-1
> parameter
[1] 4
> chisq.test(x,p=prob)$parameter
df
 4
> p.value<-1-pchisq(c,parameter)
> p.value
[1] 2.684530e-11
> chisq.test(x,p=prob)$p.value
[1] 2.684534e-11
> osservati
[1] 100 110  80  55  14
> chisq.test(x,p=prob)$observed
[1] 100 110  80  55  14
> attesi
[1] 104.11  75.39  61.03  61.03  57.44
> chisq.test(x,p=prob)$expected
[1] 104.11  75.39  61.03  61.03  57.44
> residui<-(osservati-attesi)/sqrt(attesi)
> residui
[1] -0.4028057  3.9860682  2.4282626 -0.7718726 -5.7316888
> chisq.test(x,p=prob)$residuals
[1] -0.4028057  3.9860682  2.4282626 -0.7718726 -5.7316888

> x
[1] 89 37 30 28  2
> n<-sum(x)
```

```
> n
[1] 186
> prob
[1] 0.40 0.20 0.20 0.15 0.05
> # le probabilità sommano ad 1
> k<-length(x)
> k
[1] 5
> osservati<-x
> attesi<-n*prob
> c<-sum((osservati-attesi)**2/attesi)
> c
[1] 9.990143
> chisq.test(x,p=prob)$statistic
X-squared
 9.990143
> parameter<-k-1
> parameter
[1] 4
> chisq.test(x,p=prob)$parameter
df
 4
> p.value<-1-pchisq(c,parameter)
> p.value
[1] 0.04059404
> chisq.test(x,p=prob)$p.value
[1] 0.04059404
> osservati
[1] 89 37 30 28  2
> chisq.test(x,p=prob)$observed
[1] 89 37 30 28  2
> attesi
[1] 74.4 37.2 37.2 27.9  9.3
> chisq.test(x,p=prob)$expected
[1] 74.4 37.2 37.2 27.9  9.3
> residui<-(osservati-attesi)/sqrt(attesi)
> residui
[1]  1.69264697 -0.03279129 -1.18048650  0.01893206 -2.39376430
> chisq.test(x,p=prob)$residuals
[1]  1.69264697 -0.03279129 -1.18048650  0.01893206 -2.39376430
```

Parte IV

Modelli Lineari

Capitolo 13

Regressione lineare semplice

13.1 Simbologia

$$y_i = \beta_1 + \beta_2 x_i + \varepsilon_i \quad \forall\, i = 1, 2, \ldots, n \qquad \varepsilon \sim N(0, \sigma^2 I_n)$$

- variabile dipendente: y
- matrice del modello di dimensione $n \times 2$: X
- numero di parametri da stimare e rango della matrice del modello: 2
- numero di unità: n
- i-esima riga della matrice del modello : $X_i = (1, x_i) \quad \forall\, i = 1, 2, \ldots, n$
- matrice di proiezione di dimensione $n \times n$: $H = X\,(X^T X)^{-1}\,X^T$
- matrice identità di dimensione $n \times n$: I_n
- devianza residua: $RSS = \sum_{i=1}^{n} e_i^2 = y^T\,(I_n - H)\,y$
- stima di σ^2: $s^2 = RSS\,/\,(n-2)$
- gradi di libertà della devianza residua: $n-2$
- stima di σ^2 tolta la i-esima unità: $s^2_{-i} = s^2\left(1 + \frac{1 - rstandard_i^2}{n-3}\right) = s^2\left(1 + \frac{rstudent_i^2 - 1}{n-2}\right)^{-1} \quad \forall\, i = 1, 2, \ldots, n$
- stime OLS: $\hat{\beta} = \left(\hat{\beta}_1, \hat{\beta}_2\right)^T$
- stima OLS intercetta: $\hat{\beta}_1 = \bar{y} - \frac{s_{x\,y}}{s_x^2}$
- stima OLS coefficiente angolare: $\hat{\beta}_2 = \frac{s_{x\,y}}{s_x^2}$
- standard error delle stime OLS: $s_{\hat{\beta}} = s\,\sqrt{\mathrm{diag}((X^T X)^{-1})}$
- standard error della stima OLS intercetta: $s_{\hat{\beta}_1} = s\,\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n \sum_{i=1}^{n} (x_i - \bar{x})^2}}$
- standard error della stima OLS coefficiente angolare: $s_{\hat{\beta}_2} = s\,\sqrt{\frac{1}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}$
- covarianza tra le stime OLS: $s_{\hat{\beta}_1\,\hat{\beta}_2} = -s^2\,\frac{\bar{x}}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$
- t-values delle stime OLS: $t_{\hat{\beta}} = \hat{\beta}\,/\,s_{\hat{\beta}}$
- residui: $e = (I_n - H)\,y$
- residui standard: $rstandard_i = \frac{e_i}{s\,\sqrt{1 - h_i}} \quad \forall\, i = 1, 2, \ldots, n$
- residui studentizzati: $rstudent_i = \frac{e_i}{s_{-i}\,\sqrt{1 - h_i}} \quad \forall\, i = 1, 2, \ldots, n$
- valori fittati: $\hat{y} = H\,y$
- valori di leva: $h = \mathrm{diag}(H)$
- stime OLS tolta la i-esima unità: $\hat{\beta}_{(-i)} \quad \forall\, i = 1, 2, \ldots, n$

- correlazione tra le stime OLS: $r_{\hat{\beta}_1 \hat{\beta}_2} = \frac{s_{\hat{\beta}_1 \hat{\beta}_2}}{s_{\hat{\beta}_1} s_{\hat{\beta}_2}}$
- devianza residua modello nullo: $RSS_{nullo} = \sum_{i=1}^{n} (y_i - \bar{y})^2 = (y - \bar{y})^T (y - \bar{y})$
- indice di determinazione: $R^2 = 1 - RSS \,/\, RSS_{nullo}$
- indice di determinazione aggiustato: $R^2_{adj} = 1 - \frac{RSS \,/\, (n-2)}{RSS_{nullo} \,/\, (n-1)} = 1 - (1 - R^2) \left(\frac{n-1}{n-2}\right)$
- valore noto del regressore per la previsione: x_0
- log-verosimiglianza normale: $\hat{\ell} = -n \left[\log(2\,\pi) + \log\left(RSS \,/\, n\right) + 1\right] / 2$
- distanza di *Cook*: $cd_i = \frac{h_i \, rstandard_i^2}{2\,(1-h_i)} = \frac{e_i^2}{2\,s^2} \frac{h_i}{(1-h_i)^2} \quad \forall\, i = 1, 2, \ldots, n$
- covratio: $cr_i = (1 - h_i)^{-1} \left(1 + \frac{rstudent_i^2 - 1}{n-2}\right)^{-2} = (1 - h_i)^{-1} \left(\frac{s_{-i}}{s}\right)^4 \quad \forall i = 1, 2, \ldots, n$

13.2 Stima

lm()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione lineare con una variabile esplicativa ed n unità
  `x = T / F` matrice del modello
  `y = T / F` variabile dipendente

- **Significato:** analisi di regressione lineare
- **Output:**

  `coefficients` stime OLS
  `residuals` residui
  `rank` rango della matrice del modello
  `fitted.values` valori fittati
  `df.residual` gradi di libertà della devianza residua
  `x` matrice del modello
  `y` variabile dipendente

- **Formula:**

  `coefficients`
  $$\hat{\beta}$$
  `residuals`
  $$e$$
  `rank`
  $$2$$
  `fitted.values`
  $$\hat{y}$$
  `df.residual`
  $$n - 2$$
  `x`
  $$X$$
  `y`
  $$y$$

- **Esempio:**

```
> lm(formula=y~x,x=T,y=T)
```

- **Osservazioni 1:** Il modello nullo di regressione lineare si ottiene attraverso il comando `lm(formula=y~ 1)`.
- **Osservazioni 2:** L'istruzione `lm(formula=y~ x)` è equivalente a scrivere il comando `lm(formula=y~ X-1)`.
- **Osservazioni 3:** L'istruzione `lm(formula=y~ x)` è equivalente a scrivere il comando `lm(formula=y~ 1+x)`.

summary.lm()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità
  `correlation = T / F` correlazione tra le stime OLS

- **Significato:** analisi di regressione lineare
- **Output:**

  `residuals` residui
  `coefficients` stima puntuale, standard error, t-value, p-value
  `sigma` stima di σ
  `r.squared` indice di determinazione
  `adj.r.squared` indice di determinazione aggiustato
  `fstatistic` valore empirico della statistica F, df numeratore, df denominatore
  `cov.unscaled` matrice di covarianza delle stime OLS non scalata per σ^2
  `correlation` matrice di correlazione tra le stime OLS

- **Formula:**

  `residuals`
  $$e$$
  `coefficients`
  $$\hat{\beta}_j \quad s_{\hat{\beta}_j} \quad t_{\hat{\beta}_j} \quad p\text{-value} = 2\,P(t_{n-2} \leq -|\,t_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1,\,2$$
  `sigma`
  $$s$$
  `r.squared`
  $$R^2$$
  `adj.r.squared`
  $$R^2_{adj}$$
  `fstatistic`
  $$Fvalue = \frac{(RSS_{nullo} - RSS)}{RSS\,/\,(n-2)} \qquad 1 \qquad n-2$$
  `cov.unscaled`
  $$(X^T\,X)^{-1}$$
  `correlation`
  $$r_{\hat{\beta}_1\,\hat{\beta}_2}$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> summary.lm(object=modello,correlation=T)
```

vcov()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

- **Significato:** matrice di covarianza delle stime OLS
- **Formula:**
$$s^2\,(X^T\,X)^{-1}$$
- **Esempio:**

```
> modello<-lm(formula=y~x)
> vcov(object=modello)
```

lm.fit()

- **Package:** `stats`
- **Parametri:**

  `x` matrice del modello
  `y` variabile dipendente

- **Significato:** analisi di regressione lineare
- **Output:**

  `coefficients` stime OLS
  `residuals` residui
  `rank` rango della matrice del modello
  `fitted.values` valori fittati
  `df.residual` gradi di libertà della devianza residua

- **Formula:**

  `coefficients`
  $$\hat{\beta}$$
  `residuals`
  $$e$$
  `rank`
  $$2$$
  `fitted.values`
  $$\hat{y}$$
  `df.residual`
  $$n-2$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> X<-model.matrix(object=modello)
> lm.fit(x=X,y)
```

lsfit()

- **Package:** `stats`
- **Parametri:**

  `x` matrice del modello
  `y` variabile dipendente

- **Significato:** analisi di regressione lineare
- **Output:**

  `coefficients` stime OLS
  `residuals` residui

- **Formula:**

  `coefficients`
  $$\hat{\beta}$$
  `residuals`
  $$e$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> X<-model.matrix(object=modello)
> lsfit(x=X,y,intercept=F)
```

confint()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

  `parm` parametri del modello su cui calcolare l'intervallo di confidenza

  `level` livello di confidenza $1 - \alpha$

- **Significato:** intervallo di confidenza per le stime OLS
- **Formula:**

$$\hat{\beta}_j \mp t_{1-\alpha\,/\,2,\,n-2}\, s_{\hat{\beta}_j} \quad \forall j = 1, 2$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> confint(object=modello,parm=c(1,2),level=0.95)
```

coef()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

- **Significato:** stime OLS
- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> coef(object=modello)
```

boxcox()

- **Package:** `MASS`
- **Parametri:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

  `lambda` parametro di trasformazione λ

- **Significato:** modello trasformato secondo *Box–Cox*
- **Output:**

  `x` valore del parametro λ

  `y` funzione di verosimiglianza $L(\lambda)$ da minimizzare in λ

- **Formula:**

  `x`

$$\lambda$$

`y`

$$L(\lambda) = -\frac{n}{2}\log\left(RSS_{t_\lambda(y)}\right) + (\lambda - 1)\sum_{i=1}^{n}\log(y_i)$$

$$\text{dove} \quad t_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{se } \lambda \neq 0 \\ \log(y) & \text{se } \lambda = 0 \end{cases}$$

$RSS_{t_\lambda(y)}$ rappresenta il valore di RSS per il modello che presenta $t_\lambda(y)$ come variabile dipendente.

- **Esempio:**

```
> modello<-lm(formula=y~x)
> boxcox(object=modello,lambda=1.2,plotit=F)

> modello<-lm(formula=y~x)
> boxcox(object=modello,lambda=c(1.2,2.2,3.7,4.1),plotit=F)
```

fitted()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

- **Significato:** valori fittati
- **Formula:**

$$\hat{y}$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> fitted(object=modello)
```

predict.lm()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

  `newdata` il valore di x_0

  `se.fit = T / F` standard error delle stime

  `scale` stima s^* di σ

  `df` il valore *df* dei gradi di libertà

  `interval = confidence / prediction` intervallo di confidenza o previsione

  `level` livello di confidenza $1 - \alpha$

- **Significato:** intervallo di confidenza o di previsione
- **Output:**

  `fit` valore previsto ed intervallo di confidenza

  `se.fit` standard error delle stime

  `df` il valore *df* dei gradi di libertà

  `residual.scale` stima s^* di σ

- **Formula:**

  `fit`

  `interval = confidence`

  $$\beta_1 + \beta_2 x_0 \qquad \beta_1 + \beta_2 x_0 \mp t_{1-\alpha\,/\,2,\,df}\; s^* \sqrt{\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}$$

  `interval = prediction`

  $$\beta_1 + \beta_2 x_0 \qquad \beta_1 + \beta_2 x_0 \mp t_{1-\alpha\,/\,2,\,df}\; s^* \sqrt{1 + \frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}$$

  `se.fit`

  $$s^* \sqrt{\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}$$

  `df`

  $$df \;=\; n - 2$$

  `residual.scale`

  $$s^*$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> predict.lm(object=modello,newdata=data.frame(x=1.3),interval="prediction",level=0.95,se.fit=T)
```

- **Osservazioni 1:** Per il calcolo dell'intervallo classico di confidenza o previsione impostare `scale =` s e `df =` $n - 2$.

- **Osservazioni 2:** Per il calcolo dell'intervallo asintotico di confidenza o previsione impostare `scale =` s e `df = Inf`.

13.3 Adattamento

logLik()

- **Package:** `stats`

- **Parametri:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

- **Significato:** log-verosimiglianza normale

- **Formula:**

  $$\hat{\ell}(\mu,\, \sigma^2)$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> logLik(object=modello)
```

durbin.watson()

- **Package:** `car`
- **Parametri:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità
- **Significato:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

  `dw` valore empirico della statistica D–W
- **Formula:**

  `dw`

$$\frac{\sum_{i=2}^{n} (e_i - e_{i-1})^2}{RSS}$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> durbin.watson(model=modello)
```

AIC()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità
- **Significato:** indice AIC
- **Formula:**

$$-2\,\hat{\ell} + 6$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> AIC(object=modello)
```

extractAIC()

- **Package:** `stats`
- **Parametri:**

  `fit` modello di regressione lineare con una variabile esplicativa ed n unità
- **Significato:** numero di parametri del modello ed indice AIC generalizzato
- **Formula:**

$$2 \qquad n \log(RSS\,/\,n) + 4$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> extractAIC(fit=modello)
```

deviance()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

- **Significato:** devianza residua
- **Formula:**

$$RSS$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> deviance(object=modello)
```

13.4 Diagnostica

ls.diag()

- **Package:** `stats`
- **Parametri:**

  `ls.out` modello di regressione lineare con una variabile eplicativa ed n unità

- **Significato:** analisi di regressione lineare
- **Output:**

  `std.dev` stima di σ

  `hat` valori di leva

  `std.res` residui standard

  `stud.res` residui studentizzati

  `cooks` distanza di *Cook*

  `dfits` dfits

  `correlation` matrice di correlazione tra le stime OLS

  `std.err` standard error delle stime OLS

  `cov.scaled` matrice di covarianza delle stime OLS

  `cov.unscaled` matrice di covarianza delle stime OLS non scalata per σ^2

- **Formula:**

  `std.dev`

$$s$$

  `hat`

$$h$$

  `std.res`

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

  `stud.res`

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

  `cooks`

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

  `dfits`

$$rstudent_i \sqrt{\frac{h_i}{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$$

`correlation`

$$r_{\hat{\beta}_1\,\hat{\beta}_2}$$

`std.err`

$$s_{\hat{\beta}_j} \quad \forall\, j\, =\, 1,$$

`cov.scaled`

$$s^2\,(X^T\,X)^{-1}$$

`cov.unscaled`

$$(X^T\,X)^{-1}$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> ls.diag(ls.out=modello)
```

cooks.distance()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Significato:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i\, =\, 1,\, 2,\, \ldots,\, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> cooks.distance(model=modello)
```

rstandard()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Significato:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i\, =\, 1,\, 2,\, \ldots,\, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> rstandard(model=modello)
```

rstudent()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità
- **Significato:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> rstudent(model=modello)
```

lmwork()

- **Package:** `MASS`
- **Parametri:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità
- **Significato:** diagnostica di regressione
- **Output:**

  `stdedv` stima di σ

  `stdres` residui standard

  `studres` residui studentizzati
- **Formula:**

  `stdedv`

$$s$$

  `stdres`

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

  `studres`

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> lmwork(object=modello)
```

dffits()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità
- **Significato:** dffits
- **Formula:**

$$rstudent_i \sqrt{\frac{h_i}{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> dffits(model=modello)
```

covratio()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità
- **Significato:** covratio
- **Formula:**

$$cr_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> covratio(model=modello)
```

lm.influence()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità
- **Significato:** diagnostica di regressione
- **Output:**

  `hat` valori di leva

  `coefficients` differenza tra le stime OLS eliminando una unità

  `sigma` stima di σ eliminando una unità

  `wt.res` residui
- **Formula:**

  `hat`

$$h$$

  `coefficients`

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = e_i \left(1 - h_i\right)^{-1} \left(X^T X\right)^{-1}_j X_i^T \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2$$

  `sigma`

$$s_{-i} \quad \forall\, i = 1, 2, \ldots, n$$

  `wt.res`

$$e$$

- **Esempio:**

```
> lm.influence(model=lm(formula=y~x))
```

residuals()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione lineare con una variabile esplicativa ed n unità
- **Significato:** residui
- **Formula:**

$$e$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> residuals(modello)
```

df.residual()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

- **Significato:** gradi di libertà della devianza residua
- **Formula:**

$$n - 2$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> df.residual(object=modello)
```

hatvalues()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Significato:** valori di leva
- **Formula:**

$$h$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> hatvalues(model=modello)
```

dfbeta()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Significato:** dfbeta
- **Formula:**

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = e_i \left(1 - h_i\right)^{-1} \left(X^T X\right)^{-1}_j X^T_i \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> dfbeta(model=modello)
```

dfbetas()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Significato:** dfbetas
- **Formula:**

$$\frac{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}{s_{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}} = \frac{e_i\,(1-h_i)^{-1}\,(X^T\,X)^{-1}_j\,X^T_i}{s_{-i}\,\sqrt{(X^T\,X)^{-1}_{j,\,j}}} \quad \forall i = 1,\,2,\,\ldots,\,n \quad \forall j = 1,\,2$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> dfbetas(model=modello)
```

outlier.test.lm()

- **Package:** `car`
- **Parametri:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Significato:** test sugli *outliers*
- **Output:**

  `test` massimo residuo studentizzato assoluto, gradi di libertà, p-value

- **Formula:**

  `test`

$$t = \max_i(\,|\,rstudent_i\,|\,) \quad n-3 \quad p\text{-value} = 2\,P(\,t_{n-3} \leq -|\,t\,|\,) \qquad \forall\, i = 1,\,2,\,\ldots\,,n$$

- **Esempio:**

```
> modello<-lm(formula=y~x)
> outlier.test.lm(model=modello)
```

Capitolo 14

Regressione lineare multipla

14.1 Simbologia

$$y_i = \beta_1 + \beta_2 x_{i1} + \beta_3 x_{i2} + \cdots + \beta_k x_{ik-1} + \varepsilon_i \quad \forall\, i = 1, 2, \ldots, n \qquad \varepsilon \sim N(0, \sigma^2 I_n)$$

- variabile dipendente: y
- matrice del modello di dimensione $n \times k$: X
- numero di parametri da stimare e rango della matrice del modello: k
- numero di unità: n
- i-esima riga della matrice del modello : $X_i = (1, x_{i1}, x_{i2}, \ldots, x_{ik-1}) \quad \forall\, i = 1, 2, \ldots, n$
- matrice di proiezione di dimensione $n \times n$: $H = X\,(X^T X)^{-1}\,X^T$
- matrice identità di dimensione $n \times n$: I_n
- devianza residua: $RSS = \sum_{i=1}^{n} e_i^2 = y^T\,(I_n - H)\,y$
- stima di σ^2: $s^2 = RSS\,/\,(n-k)$
- gradi di libertà della devianza residua: $n-k$
- stima di σ^2 tolta la i-esima unità: $s^2_{-i} = s^2\left(1 + \frac{1 - rstandard_i^2}{n-k-1}\right) = s^2\left(1 + \frac{rstudent_i^2 - 1}{n-k}\right)^{-1} \quad \forall\, i = 1, 2, \ldots, n$
- stime OLS: $\hat{\beta} = (X^T X)^{-1}\,X^T\,y$
- standard error delle stime OLS: $s_{\hat{\beta}} = s\,\sqrt{\mathrm{diag}((X^T X)^{-1})}$
- t-values delle stime OLS: $t_{\hat{\beta}} = \hat{\beta}\,/\,s_{\hat{\beta}}$
- residui: $e = (I_n - H)\,y$
- residui standard: $rstandard_i = \frac{e_i}{s\,\sqrt{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$
- residui studentizzati: $rstudent_i = \frac{e_i}{s_{-i}\,\sqrt{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$
- valori fittati: $\hat{y} = H\,y$
- valori di leva: $h = \mathrm{diag}(H)$
- stime OLS tolta la i-esima unità: $\hat{\beta}_{(-i)} \quad \forall\, i = 1, 2, \ldots, n$
- correlazione tra le stime OLS: $r_{\hat{\beta}_i\,\hat{\beta}_j} = \frac{s^2\,(X^T X)^{-1}_{(i,\,j)}}{s_{\hat{\beta}_i}\,s_{\hat{\beta}_j}} \quad \forall\, i, j = 1, 2, \ldots, k$
- devianza residua modello nullo: $RSS_{nullo} = \sum_{i=1}^{n} (y_i - \bar{y})^2 = (y - \bar{y})^T\,(y - \bar{y})$
- indice di determinazione: $R^2 = 1 - RSS\,/\,RSS_{nullo}$
- indice di determinazione aggiustato: $R^2_{adj} = 1 - \frac{RSS\,/\,(n-k)}{RSS_{nullo}\,/\,(n-1)} = 1 - (1 - R^2)\left(\frac{n-1}{n-k}\right)$
- valore noto dei regressori per la previsione: $x_0^T = (1, x_{01}, x_{02}, \ldots, x_{0k-1})$

- log-verosimiglianza normale: $\hat{\ell} = -n\left[\log(2\,\pi) + \log\left(RSS\,/\,n\right) + 1\right]\,/\,2$
- distanza di *Cook*: $cd_i = \frac{h_i\,rstandard_i^2}{k\,(1-h_i)} = \frac{e_i^2}{k\,s^2}\,\frac{h_i}{(1-h_i)^2} \quad \forall\, i = 1,\,2,\,\ldots,\,n$
- covratio: $cr_i = (1-h_i)^{-1}\left(1 + \frac{rstudent_i^2 - 1}{n-k}\right)^{-k} = (1-h_i)^{-1}\left(\frac{s_{-i}}{s}\right)^{2\,k} \quad \forall i = 1,\,2,\,\ldots,\,n$

14.2 Stima

lm()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `x = T / F` matrice del modello

  `y = T / F` variabile dipendente

- **Significato:** analisi di regressione lineare
- **Output:**

  `coefficients` stime OLS

  `residuals` residui

  `rank` rango della matrice del modello

  `fitted.values` valori fittati

  `df.residual` gradi di libertà della devianza residua

  `x` matrice del modello

  `y` variabile dipendente

- **Formula:**

  `coefficients`
  $$\hat{\beta}$$

  `residuals`
  $$e$$

  `rank`
  $$k$$

  `fitted.values`
  $$\hat{y}$$

  `df.residual`
  $$n-k$$

  `x`
  $$X$$

  `y`
  $$y$$

- **Esempio:**

```
> lm(formula=y~x1+x2+x3,x=T,y=T)
```

- **Osservazioni 1:** Il modello nullo di regressione lineare si ottiene attraverso il comando `lm(formula=y~ 1)`.
- **Osservazioni 2:** L'istruzione `update(object=y~ x1+x2,formula=.~ .+x3)` è equivalente a scrivere il comando `lm(formula=y~ x1+x2+x3)`.
- **Osservazioni 3:** In seguito ad una modifica come ad esempio `x1[3]<-1.2`, conviene adoperare il comando `update(modello)` anziché ripetere nuovamente `modello<-lm(y~ x1+x2+x3)`.

- **Osservazioni 4:** L'operatore `I()` permette di poter modellare regressioni lineari polinomiali. Per un polinomio di terzo grado occorre scrivere `lm(formula=y~ x+I(x**2)+I(x**3))`.
- **Osservazioni 5:** Per regressioni polinomiali ortogonali occorre usare il comando `poly()`. Per un polinomio ortogonale di quarto grado occorre scrivere `lm(formula=y~ poly(x,degree=4))`.
- **Osservazioni 6:** L'istruzione `lm(formula=y~ x1+x2+x3)` è equivalente a scrivere il comando `lm(formula=y~ X-1)`.
- **Osservazioni 7:** L'istruzione `lm(formula=y~ x1+x2+x3)` è equivalente a scrivere il comando `lm(formula=y~ 1+x1+x2+x3)`.

summary.lm()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `correlation = T / F` correlazione tra le stime OLS

- **Significato:** analisi di regressione lineare
- **Output:**

  `residuals` residui

  `coefficients` stima puntuale, standard error, t-value, p-value

  `sigma` stima di σ

  `r.squared` indice di determinazione

  `adj.r.squared` indice di determinazione aggiustato

  `fstatistic` valore empirico della statistica F, df numeratore, df denominatore

  `cov.unscaled` matrice di covarianza delle stime OLS non scalata per σ^2

  `correlation` matrice di correlazione tra le stime OLS

- **Formula:**

  `residuals`

  $$e$$

  `coefficients`

  $$\hat{\beta}_j \quad s_{\hat{\beta}_j} \quad t_{\hat{\beta}_j} \quad p\text{-value} = 2\,P(t_{n-k} \leq -|\,t_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1, 2, \ldots, k$$

  `sigma`

  $$s$$

  `r.squared`

  $$R^2$$

  `adj.r.squared`

  $$R^2_{adj}$$

  `fstatistic`

  $$Fvalue = \frac{\left(RSS_{nullo} - RSS\right) / (k-1)}{RSS\,/\,(n-k)} \qquad k-1 \qquad n-k$$

  `cov.unscaled`

  $$(X^T\,X)^{-1}$$

  `correlation`

  $$r_{\hat{\beta}_i\,\hat{\beta}_j} \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> summary.lm(object=modello,correlation=T)
```

vcov()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** matrice di covarianza delle stime OLS
- **Formula:**

$$s^2 \, (X^T \, X)^{-1}$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> vcov(object=modello)
```

lm.fit()

- **Package:** `stats`
- **Parametri:**

  `x` matrice del modello

  `y` variabile dipendente

- **Significato:** analisi di regressione lineare
- **Output:**

  `coefficients` stime OLS

  `residuals` residui

  `rank` rango della matrice del modello

  `fitted.values` valori fittati

  `df.residual` gradi di libertà della devianza residua

- **Formula:**

  `coefficients`

$$\hat{\beta}$$

  `residuals`

$$e$$

  `rank`

$$k$$

  `fitted.values`

$$\hat{y}$$

  `df.residual`

$$n-k$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> X<-model.matrix(object=modello)
> lm.fit(x=X,y)
```

lsfit()

- **Package:** `stats`
- **Parametri:**

  `x` matrice del modello

  `y` variabile dipendente

- **Significato:** analisi di regressione lineare
- **Output:**

  `coefficients` stime OLS

  `residuals` residui

- **Formula:**

  `coefficients`

  $$\hat{\beta}$$

  `residuals`

  $$e$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> X<-model.matrix(object=modello)
> lsfit(x=X,y,intercept=F)
```

confint()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `parm` parametri del modello su cui calcolare l'intervallo di confidenza

  `level` livello di confidenza $1-\alpha$

- **Significato:** intervallo di confidenza per le stime OLS
- **Formula:**

$$\hat{\beta}_j \mp t_{1-\alpha/2,\,n-k}\, s_{\hat{\beta}_j} \quad \forall j = 1, 2, \ldots, k$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> confint(object=modello,parm=c(1,2,3),level=0.95)
```

Confint()

- **Package:** `Rcmdr`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `parm` parametri del modello su cui calcolare l'intervallo di confidenza

  `level` livello di confidenza $1-\alpha$

- **Significato:** intervallo di confidenza per le stime OLS
- **Formula:**

$$\hat{\beta}_j \mp t_{1-\alpha/2,\,n-k}\, s_{\hat{\beta}_j} \quad \forall j = 1, 2, \ldots, k$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> Confint(object=modello,parm=c(1,2,3),level=0.95)
```

coef()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** stime OLS
- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> coef(object=modello)
```

coefficients()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** stime OLS
- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> coefficients(object=modello)
```

coeftest()

- **Package:** `lmtest`
- **Parametri:**

  `x` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `df = NULL / Inf` significatività delle stime effettuata con la variabile casuale t oppure Z

- **Significato:** stima puntuale, standard error, t-value, p-value
- **Formula:**

`df = NULL`

$$\hat{\beta}_j \quad s_{\hat{\beta}_j} \quad t_{\hat{\beta}_j} \quad p\text{-value} = 2\,P(t_{n-k} \leq -|\,t_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1, 2, \ldots, k$$

`df = Inf`

$$\hat{\beta}_j \quad s_{\hat{\beta}_j} \quad t_{\hat{\beta}_j} \quad p\text{-value} = 2\,\Phi\left(-|\,t_{\hat{\beta}_j}\,|\right) \qquad \forall\, j = 1, 2, \ldots, k$$

- **Esempio:**

```
> coeftest(x=lm(formula=y~x1+x2+x3),df=NULL)

> coeftest(x=lm(formula=y~x1+x2+x3),df=Inf)
```

boxcox()

- **Package:** `MASS`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità
  `lambda` parametro di trasformazione λ
- **Significato:** modello trasformato secondo *Box–Cox*
- **Output:**

  `x` valore del parametro λ
  `y` funzione di verosimiglianza $L(\lambda)$ da minimizzare in λ
- **Formula:**

  `x`

  $$\lambda$$

  `y`

  $$L(\lambda) = -\frac{n}{2}\log\left(RSS_{t_\lambda(y)}\right) + (\lambda-1)\sum_{i=1}^{n}\log(y_i)$$

  $$\text{dove} \quad t_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{se } \lambda \neq 0 \\ \log(y) & \text{se } \lambda = 0 \end{cases}$$

  $RSS_{t_\lambda(y)}$ rappresenta il valore di RSS per il modello che presenta $t_\lambda(y)$ come variabile dipendente.
- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> boxcox(object=modello,lambda=1.2,plotit=F)

> modello<-lm(formula=y~x1+x2+x3)
> boxcox(object=modello,lambda=c(1.2,2.2,3.7,4.1),plotit=F)
```

box.cox()

- **Package:** `car`
- **Parametri:**

  `y` vettore numerico positivo di dimensione n
  `p` parametro di trasformazione λ
- **Significato:** variabile y trasformata secondo *Box–Cox*
- **Formula:**

$$t_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{se } \lambda \neq 0 \\ \log(y) & \text{se } \lambda = 0 \end{cases}$$

- **Esempio:**

```
> box.cox(y,p=0.5)
```

box.cox.var()

- **Package:** `car`
- **Parametri:**

  `y` vettore numerico positivo di dimensione n

- **Significato:** variabile y trasformata secondo *Box–Cox*
- **Formula:**

$$(\log(y \,/\, \bar{y}_G) - 1)\; y$$

- **Esempio:**

```
> box.cox.var(y)
```

bc()

- **Package:** `car`
- **Parametri:**

  `y` vettore numerico positivo di dimensione n

  `p` parametro di trasformazione λ

- **Significato:** variabile y trasformata secondo *Box–Cox*
- **Formula:**

$$t_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{se } \lambda \neq 0 \\ \log(y) & \text{se } \lambda = 0 \end{cases}$$

- **Esempio:**

```
> bc(y,p=0.5)
```

fitted()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** valori fittati
- **Formula:**

$$\hat{y}$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> fitted(object=modello)
```

fitted.values()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** valori fittati
- **Formula:**

$$\hat{y}$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> fitted.values(object=modello)
```

predict.lm()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `newdata` il valore di x_0

  `se.fit = T / F` standard error delle stime

  `scale` stima s^* di σ

  `df` il valore *df* dei gradi di libertà

  `interval = confidence / prediction` intervallo di confidenza o previsione

  `level` livello di confidenza $1-\alpha$

- **Significato:** intervallo di confidenza o di previsione
- **Output:**

  `fit` valore previsto ed intervallo di confidenza

  `se.fit` standard error delle stime

  `df` il valore *df* dei gradi di libertà

  `residual.scale` stima s^* di σ

- **Formula:**

  `fit`

  `interval = confidence`

$$x_0^T\,\hat{\beta} \qquad x_0^T\,\hat{\beta} \mp t_{1-\alpha\,/\,2,\,df}\; s^* \sqrt{x_0^T\,(X^T\,X)^{-1}\,x_0}$$

  `interval = prediction`

$$x_0^T\,\hat{\beta} \qquad x_0^T\,\hat{\beta} \mp t_{1-\alpha\,/\,2,\,df}\; s^* \sqrt{1 + x_0^T\,(X^T\,X)^{-1}\,x_0}$$

  `se.fit`

$$s^* \sqrt{x_0^T\,(X^T\,X)^{-1}\,x_0}$$

  `df`

$$df \;=\; n-k$$

  `residual.scale`

$$s^*$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> n<-length(y)
> n
[1] 13
> k<-4
> x0<-c(1,1.3,2.1,2.3)
> yhat<-as.numeric(t(x0)%*%coef(modello))
> yhat
[1] -67.63043
> new<-data.frame(x1=1.3,x2=2.1,x3=2.3)
> s<-summary(modello)$sigma
> X<-model.matrix(object=modello)
> lower<-yhat-qt(1-0.05/2,n-k)*s*sqrt(1+t(x0)%*%solve(t(X)%*%X)%*%x0)
> upper<-yhat+qt(1-0.05/2,n-k)*s*sqrt(1+t(x0)%*%solve(t(X)%*%X)%*%x0)
> c(yhat,lower,upper)
[1]  -67.63043 -108.91459  -26.34627
> res<-predict.lm(object=modello,newdata=new,interval="prediction",level=0.95,se.fit=T)
> res$fit
           fit       lwr       upr
[1,] -67.63043 -108.9146 -26.34627
> se.fit<-as.numeric(s*sqrt(t(x0)%*%solve(t(X)%*%X)%*%x0))
> se.fit
[1] 18.15023
> res$se.fit
[1] 18.15023
> s
[1] 1.904851
> res$residual.scale
[1] 1.904851
```

- **Osservazioni 1:** Per il calcolo dell'intervallo classico di confidenza o previsione impostare `scale =` s e `df =` $n-k$.
- **Osservazioni 2:** Per il calcolo dell'intervallo asintotico di confidenza o previsione impostare `scale =` s e `df = Inf`.

linear.hypothesis.lm()

- **Package:** `car`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `hypothesis.matrix` matrice C di dimensione $q \times k$ e rango pari a $q = \min(q, k)$

  `rhs` vettore b della previsione lineare di dimensione q

- **Significato:** test di ipotesi per $H_0 : C\,\beta = b$ contro $H_1 : C\,\beta \neq b$ dove C e b sono così definiti:

$$C = \begin{pmatrix} c_{1,1} & c_{1,2} & \dots & c_{1,k} \\ c_{2,1} & c_{2,2} & \dots & c_{2,k} \\ \vdots & \vdots & \vdots & \vdots \\ c_{q,1} & c_{q,2} & \dots & c_{q,k} \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_q \end{pmatrix}$$

- **Output:**

  `Res.Df` gradi di libertà della devianza residua

  `RSS` devianza residua

  `Df` gradi di libertà della devianza relativa all'ipotesi nulla H_0

  `Sum of Sq` devianza relativa all'ipotesi nulla H_0

  `F` valore empirico della statistica F

  `Pr(>F)` p-value

- **Formula:**

  `Res.Df`

  $$n-k \qquad n-k+q$$

  `RSS`

  $$RSS \qquad RSS + \left(b - C\,\hat{\beta}\right)^T \left[C\,\left(X^T\,X\right)^{-1}\,C^T\right]^{-1}\,\left(b - C\,\hat{\beta}\right)$$

  `Df`

  $$-q$$

  `Sum of Sq`

  $$-\left(b - C\,\hat{\beta}\right)^T \left[C\,\left(X^T\,X\right)^{-1}\,C^T\right]^{-1}\,\left(b - C\,\hat{\beta}\right)$$

  `F`

  $$Fvalue = \frac{\left[\left(b - C\,\hat{\beta}\right)^T \left[C\,\left(X^T\,X\right)^{-1}\,C^T\right]^{-1}\,\left(b - C\,\hat{\beta}\right)\right] / q}{RSS\,/\,(n-k)}$$

  `Pr(>F)`

  $$P(F_{q,\,n-k} \geq Fvalue)$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> k<-4
> q<-2
> C
     [,1] [,2] [,3] [,4]
[1,]    1    3  5.0  2.3
[2,]    2    4  1.1  4.3
> b
     [,1]
[1,]  1.1
[2,]  2.3
> linear.hypothesis.lm(model=modello,hypothesis.matrix=C,rhs=b)
```

lht()

- **Package:** `car`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `hypothesis.matrix` matrice C di dimensione $q \times k$ e rango pari a $q = \min(q,\,k)$

  `rhs` vettore b della previsione lineare di dimensione q

- **Significato:** test di ipotesi per $H_0 : C\,\beta = b$ contro $H_1 : C\,\beta \neq b$ dove C e b sono così definiti:

$$C = \begin{pmatrix} c_{1,\,1} & c_{1,\,2} & \dots & c_{1,\,k} \\ c_{2,\,1} & c_{2,\,2} & \dots & c_{2,\,k} \\ \vdots & \vdots & \vdots & \vdots \\ c_{q,\,1} & c_{q,\,2} & \dots & c_{q,\,k} \end{pmatrix} \qquad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_q \end{pmatrix}$$

- **Output:**

  `Res.Df` gradi di libertà della devianza residua

  `RSS` devianza residua

  `Df` gradi di libertà della devianza relativa all'ipotesi nulla H_0

  `Sum of Sq` devianza relativa all'ipotesi nulla H_0

  `F` valore empirico della statistica F

  `Pr(>F)` p-value

- **Formula:**

  `Res.Df`

  $$n-k \qquad n-k+q$$

  `RSS`

  $$RSS \qquad RSS + \left(b - C\,\hat{\beta}\right)^T \left[C\,\left(X^T\,X\right)^{-1}\,C^T\right]^{-1}\,\left(b - C\,\hat{\beta}\right)$$

  `Df`

  $$-q$$

  `Sum of Sq`

  $$-\left(b - C\,\hat{\beta}\right)^T \left[C\,\left(X^T\,X\right)^{-1}\,C^T\right]^{-1}\,\left(b - C\,\hat{\beta}\right)$$

  `F`

  $$Fvalue = \frac{\left[\left(b - C\,\hat{\beta}\right)^T \left[C\,\left(X^T\,X\right)^{-1}\,C^T\right]^{-1}\,\left(b - C\,\hat{\beta}\right)\right] / q}{RSS\,/\,(n-k)}$$

  `Pr(>F)`

  $$P(F_{q,\,n-k} \geq Fvalue)$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> k<-4
> q<-2
> C
     [,1] [,2] [,3] [,4]
[1,]    1    3  5.0  2.3
[2,]    2    4  1.1  4.3
> b
     [,1]
[1,]  1.1
[2,]  2.3
> lht(model=modello,hypothesis.matrix=C,rhs=b)
```

lm.ridge()

- **Package:** `MASS`
- **Parametri:**

  `formula` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `lambda` valore del parametro λ

- **Significato:** Ridge–Regression
- **Output:**

  `coef` stime

  `scales` scarto quadratico medio delle $k-1$ variabili esplicative

  `lambda` λ

  `ym` media della variabile dipendente

  `xm` media delle $k-1$ variabili esplicative

  `GCV` i valori di λ e GCV

  `kHKB` $kHKB$

  `kLW` kLW

- **Formula:**

  `coef`

  $$V\,(\,D^2\,+\,\lambda\,I_{k-1}\,)^{-1}\,D\,U^T\,(y-\bar{y})$$

`scales`

$$\sigma_{x_j} \quad \forall\, j \,=\, 1,\, 2,\, \ldots\,,\, k-1$$

`lambda`

$$\lambda$$

`ym`

$$\bar{y}$$

`xm`

$$\bar{x}_j \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

`GCV`

$$\lambda \qquad \frac{(y-\bar{y})^T\,(\,I_n - U\,D\,(\,D^2 + \lambda\,I_{k-1}\,)^{-1}\,D\,U^T\,)^2\,(y-\bar{y})}{\left(n - \sum_{i=1}^{k-1}\,\frac{D^2_{i,\,i}}{\lambda + D^2_{i,\,i}}\right)^2}$$

`kHKB`

$$\frac{k-3}{n-k}\,\frac{(y-\bar{y})^T\,(\,I_n - U\,U^T\,)\,(y-\bar{y})}{(y-\bar{y})^T\,U\,D^{-2}\,U^T\,(y-\bar{y})}$$

`kLW`

$$\frac{n\,(k-3)}{n-k}\,\frac{(y-\bar{y})^T\,(\,I_n\, - U\,U^T\,)\,(y-\bar{y})}{(y-\bar{y})^T\,U\,U^T\,(y-\bar{y})}$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> lm.ridge(formula=modello,lambda=1.2)
```

- **Osservazioni 1:** La matrice del modello X viene privata della prima colonna (intercetta) e poi trasformata nella matrice standardizzata Z. Successivamente viene applicata la fattorizzazione ai valori singolari $Z = U\,D\,V^T$ mediante il comando `svd()`.

- **Osservazioni 2:** I parametri stimati sono $k-1$ e non k (modello senza intercetta).

14.3 Adattamento

logLik()

- **Package:** `stats`

- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** log-verosimiglianza normale

- **Formula:**

$$\hat{\ell}(\mu,\, \sigma^2)$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> logLik(object=modello)
```

durbin.watson()

- **Package:** `car`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

  `dw` valore empirico della statistica D–W

- **Formula:**

  `dw`

$$\frac{\sum_{i=2}^{n}(e_i - e_{i-1})^2}{RSS}$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> durbin.watson(model=modello)
```

AIC()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell} + 2\,(k+1)$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> AIC(object=modello)
```

BIC()

- **Package:** `nlme`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** indice *BIC*
- **Formula:**

$$-2\,\hat{\ell} + (k+1)\,\log(n)$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> BIC(object=modello)
```

extractAIC()

- **Package:** `stats`
- **Parametri:**

  `fit` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** numero di parametri del modello ed indice AIC generalizzato
- **Formula:**

$$k \qquad n \log(RSS\,/\,n) + 2\,k$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> extractAIC(fit=modello)
```

deviance()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** devianza residua
- **Formula:**

$$RSS$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> deviance(object=modello)
```

leaps()

- **Package:** `leaps`
- **Parametri:**

  `x` matrice del modello priva della prima colonna (intercetta) di dimensione $n \times (h-1)$

  `y` variabile dipendente

- **Significato:** Best Subsets
- **Output:**

  `which` variabili selezionate

  `size` numero di parametri

  `method = r2 / adjr2 / Cp` indice R^2, R^2_{adj}, C_p

- **Formula:**

  `size`

$$k_j \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, h-1$$

  `method`

`method = r2`

Numero di esplicative	Numero di parametri	Numero di Subsets
1	$k_1 = 2$	$\binom{h-1}{1}$
2	$k_2 = 3$	$\binom{h-1}{2}$
.	.	.
.	.	.
j	$k_j = j+1$	$\binom{h-1}{j}$
.	.	.
.	.	.
$h-1$	$k_{h-1} = h$	$\binom{h-1}{h-1}$

$$R_j^2 \quad \forall\, j = 1, 2, \ldots, h-1$$

R_j^2 rappresenta il massimo R^2 tra i $\binom{h-1}{j}$ modelli di regressione con j variabili esplicative oppure k_j parametri.

`method = adjr2`

$$\begin{aligned} R^2_{adj\,j} &= 1 - \frac{RSS\,/\,(n-k_j)}{RSS_{nullo}\,/\,(n-1)} \\ &= \frac{1-k_j}{n-k_j} + \frac{n-1}{n-k_j}\,R_j^2 \qquad \forall\, j = 1, 2, \ldots, h-1 \end{aligned}$$

$R^2_{adj\,j}$ rappresenta il massimo R^2_{adj} tra i $\binom{h-1}{j}$ modelli di regressione con j variabili esplicative oppure k_j parametri.

`method = Cp`

$$\begin{aligned} Cp_j &= (n-k_{h-1})\,\frac{1-R_j^2}{1-R_{h-1}^2} + 2\,k_j - n \\ &= \left(\frac{n-k_{h-1}}{1-R_{h-1}^2} + 2\,k_j - n\right) - \frac{n-k_{h-1}}{1-R_{h-1}^2}\,R_j^2 \qquad \forall\, j = 1, 2, \ldots, h-1 \end{aligned}$$

Cp_j rappresenta il minimo Cp tra i $\binom{h-1}{j}$ modelli di regressione con j variabili esplicative oppure k_j parametri.

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> X<-model.matrix(object=modello)
> A<-X[,-1]
> leaps(x=A,y,method="adjr2",nbest=1)

> modello<-lm(formula=y~x1+x2+x3)
> X<-model.matrix(object=modello)
> A<-X[,-1]
> leaps(x=A,y,method="Cp",nbest=1)
```

- **Osservazioni 1:** Tutti i modelli contengono l'intercetta.
- **Osservazioni 2:** $R^2_{adj\,j}$ è una trasformazione lineare crescente di R_j^2.
- **Osservazioni 3:** Cp_j è una trasformazione lineare decrescente di R_j^2.

drop1()

- **Package:** `stats`
- **Parametri:**

`object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

`scale` selezione indice AIC oppure Cp

- **Significato:** Submodels
- **Output:**

`Df` differenza tra gradi di libertà

`Sum of Sq` differenza tra devianze residue

`RSS` devianza residua

`AIC` indice AIC

`Cp` indice Cp

`F value` valore empirico della statistica F

`Pr(F)` p-value

- **Formula:**

`Df`

$$\underbrace{1,\, 1,\, \ldots,\, 1}_{k-1 \text{ volte}}$$

`Sum of Sq`

$$RSS_{-x_j} - RSS \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

dove RSS_{-x_j} rappresenta la devianza residua del modello eliminata la variabile esplicativa x_j.

`RSS`

$$RSS,\, RSS_{-x_j} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

`AIC`

`scale = NULL`

$$n \log\left(RSS \,/\, n\right) + 2\,k,\; n \log\left(RSS_{-x_j} \,/\, n\right) + 2\,(k-1) \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

`Cp`

`scale =` s^2

$$k,\; (n-k)\,\frac{RSS_{-x_j}}{RSS} + 2\,(k-1) - n \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

`F value`

$$F_{-x_j} \,=\, \frac{RSS_{-x_j} - RSS}{RSS \,/\, (n-k)} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

`Pr(F)`

$$P(F_{1,\,n-k} \geq F_{-x_j}) \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

- **Esempio:**

```
> # indice AIC
> modello<-lm(formula=y~x1+x2+x3)
> drop1(object=modello,scale=NULL,test="F")

> # indice Cp
> modello<-lm(formula=y~x1+x2+x3)
> s<-summary.lm(object=modello)$sigma
> drop1(object=modello,scale=s**2,test="F")
```

add1()

- **Package:** `stats`
- **Parametri:**

  `object` modello nullo di regressione

  `scope` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `scale` selezione indice AIC oppure Cp

- **Significato:** Submodels
- **Output:**

  `Df` differenza tra gradi di libertà

  `Sum of Sq` differenza tra devianze residue

  `RSS` devianza residua

  `AIC` indice AIC

  `Cp` indice Cp

  `F value` valore empirico della statistica F

  `Pr(F)` p-value

- **Formula:**

  `Df`

  $$1$$

  `Sum of Sq`

  $$RSS_{nullo} - RSS_{x_j} \quad \forall\, j = 1, 2, \ldots, k-1$$

  RSS_{x_j} rappresenta la devianza residua del modello con la sola variabile esplicativa x_j.

  `RSS`

  $$RSS_{nullo},\, RSS_{x_j} \quad \forall\, j = 1, 2, \ldots, k-1$$

  `AIC`

  `scale = NULL`

  $$RSS_{nullo},\, n \log\left(RSS_{x_j} / n\right) + 4 \quad \forall\, j = 1, 2, \ldots, k-1$$

  `Cp`

  `scale =` s^2

  $$1,\, (n-1)\,\frac{RSS_{x_j}}{RSS_{nullo}} + 4 - n \quad \forall\, j = 1, 2, \ldots, k-1$$

  `F value`

  $$F_{x_j} = \frac{RSS_{nullo} - RSS_{x_j}}{RSS_{x_j} / (n-2)} \quad \forall\, j = 1, 2, \ldots, k-1$$

  `Pr(F)`

  $$P(F_{1,\,n-2} \geq F_{x_j}) \quad \forall\, j = 1, 2, \ldots, k-1$$

- **Esempio:**

```
> # indice AIC
> nullo<-lm(formula=y~1)
> modello<-lm(formula=y~x1+x2+x3)
> add1(object=nullo,scope=modello,scale=NULL,test="F")

> # indice Cp
> nullo<-lm(formula=y~1)
> modello<-lm(formula=y~x1+x2+x3)
> s<-summary.lm(object=nullo)$sigma
> add1(object=nullo,scope=modello,scale=s**2,test="F")
```

bptest()

- **Package:** `lmtest`
- **Parametri:**

  `formula` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `studentize = T / F` metodo di *Koenker*

- **Significato:** test di *Breusch-Pagan* per l'omoschedasticità dei residui
- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value

- **Formula:**

  `statistic`

  `studentize = T`

$$v_i = e_i^2 - RSS / n \quad \forall\, i = 1, 2, \ldots, n$$

$$c = n \frac{v^T H\, v}{v^T v}$$

  `studentize = F`

$$v_i = n\, e_i^2 / RSS - 1 \quad \forall\, i = 1, 2, \ldots, n$$

$$c = \frac{1}{2} v^T H\, v$$

  `parameter`

$$k - 1$$

  `p.value`

$$P(\chi^2_{k-1} \geq c)$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> bptest(formula=modello,studentize=T)
```

14.4 Diagnostica

ls.diag()

- **Package:** `stats`
- **Parametri:**

  `ls.out` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** analisi di regressione lineare
- **Output:**

  `std.dev` stima di σ

  `hat` valori di leva

  `std.res` residui standard

  `stud.res` residui studentizzati

  `cooks` distanza di *Cook*

`dfits` dfits

`correlation` matrice di correlazione tra le stime OLS

`std.err` standard error delle stime OLS

`cov.scaled` matrice di covarianza delle stime OLS

`cov.unscaled` matrice di covarianza delle stime OLS non scalata per σ^2

- **Formula:**

`std.dev`

$$s$$

`hat`

$$h$$

`std.res`

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

`stud.res`

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

`cooks`

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

`dfits`

$$rstudent_i \sqrt{\frac{h_i}{1 - h_i}} \quad \forall\, i = 1, 2, \ldots, n$$

`correlation`

$$r_{\hat{\beta}_i\, \hat{\beta}_j} \quad \forall\, i, j = 1, 2, \ldots, k$$

`std.err`

$$s_{\hat{\beta}_j} \quad \forall\, j = 1, 2, \ldots, k$$

`cov.scaled`

$$s^2 (X^T X)^{-1}$$

`cov.unscaled`

$$(X^T X)^{-1}$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> ls.diag(ls.out=modello)
```

cooks.distance()

- **Package:** `stats`
- **Parametri:**

`model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> cooks.distance(model=modello)
```

cookd()

- **Package:** `car`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> cookd(model=modello)
```

mahalanobis()

- **Package:** `stats`
- **Parametri:**

  `x` vettore numerico $(x_{i1}, x_{i2}, \ldots, x_{ik-1})$ di dimensione $k-1$

  `center` vettore $\bar{x}$ delle medie delle variabili indipendenti $x_1, x_2, \ldots, x_{k-1}$

  `cov` matrice S di covarianza tra le variabili indipendenti $x_1, x_2, \ldots, x_{k-1}$ di dimensione $(k-1) \times (k-1)$

- **Significato:** distanza di *Mahalanobis* al quadrato
- **Formula:**

$$MD^2 = (x - \bar{x})^T S^{-1} (x - \bar{x})$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> k<-4
> X<-model.matrix(modello)
> medie<-apply(X[,-1],MARGIN=2,FUN=mean)
> S<-cov(X[,-1])
> mahalanobis(x=X[1,-1],center=medie,cov=S,inverted=F)
```

rstandard()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> rstandard(model=modello)
```

stdres()

- **Package:** `MASS`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> stdres(object=modello)
```

rstudent()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> rstudent(model=modello)
```

studres()

- **Package:** `MASS`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> studres(object=modello)
```

lmwork()

- **Package:** `MASS`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** diagnostica di regressione
- **Output:**

  `stdedv` stima di σ

  `stdres` residui standard

  `studres` residui studentizzati

- **Formula:**

  `stdedv`

  $$s$$

  `stdres`

  $$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

  `studres`

  $$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> lmwork(object=modello)
```

dffits()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** dffits
- **Formula:**

$$rstudent_i \sqrt{\frac{h_i}{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> dffits(model=modello)
```

covratio()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** covratio
- **Formula:**

$$cr_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> covratio(model=modello)
```

lm.influence()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Significato:** diagnostica di regressione
- **Output:**

  `hat` valori di leva
  `coefficients` differenza tra le stime OLS eliminando una unità
  `sigma` stima di σ eliminando una unità
  `wt.res` residui
- **Formula:**

  `hat`

  $$h$$

  `coefficients`

  $$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = e_i \, (1 - h_i)^{-1} \, (X^T X)^{-1}_j \, X^T_i \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

  `sigma`

  $$s_{-i} \quad \forall i = 1, 2, \ldots, n$$

  `wt.res`

  $$e$$
- **Esempio:**

```
> lm.influence(model=lm(formula=y~x1+x2+x3))
```

influence()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Significato:** diagnostica di regressione
- **Output:**

  `hat` valori di leva
  `coefficients` differenza tra le stime OLS eliminando una unità
  `sigma` stima di σ eliminando una unità
  `wt.res` residui
- **Formula:**

  `hat`

  $$h$$

  `coefficients`

  $$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = e_i \, (1 - h_i)^{-1} \, (X^T X)^{-1}_j \, X^T_i \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

  `sigma`

  $$s_{-i} \quad \forall i = 1, 2, \ldots, n$$

  `wt.res`

  $$e$$
- **Esempio:**

```
> influence(model=lm(formula=y~x1+x2+x3))
```

residuals()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** residui
- **Formula:**

$$e$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> residuals(modello)
```

residuals.default()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** residui
- **Formula:**

$$e$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> residuals.default(object=modello)
```

resid()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** residui
- **Formula:**

$$e$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> resid(object=modello)
```

df.residual()

- **Package:** `stats`
- **Parametri:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** gradi di libertà della devianza residua
- **Formula:**

$$n-k$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> df.residual(object=modello)
```

hatvalues()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** valori di leva
- **Formula:**

$$h$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> hatvalues(model=modello)
```

hat()

- **Package:** `stats`
- **Parametri:**

  `x` matrice del modello

- **Significato:** valori di leva
- **Formula:**

$$h$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> X<-model.matrix(object=modello)
> hat(x=X,intercept=T)
```

dfbeta()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** dfbeta
- **Formula:**

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = e_i \left(1 - h_i\right)^{-1} \left(X^T X\right)_j^{-1} X_i^T \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> dfbeta(model=modello)
```

dfbetas()

- **Package:** `stats`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** dfbetas
- **Formula:**

$$\frac{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}{s_{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}} = \frac{e_i \left(1 - h_i\right)^{-1} \left(X^T X\right)_j^{-1} X_i^T}{s_{-i} \sqrt{\left(X^T X\right)_{j,\,j}^{-1}}} \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> dfbetas(model=modello)
```

vif()

- **Package:** `car`
- **Parametri:**

  `mod` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** variance inflation factors
- **Formula:**

$$\left(1 - R_{x_j}^2\right)^{-1} \quad \forall\, j = 1, 2, \ldots, k-1$$

  $R_{x_j}^2$ rappresenta il valore di R^2 per il modello che presenta il regressore j-esimo come variabile dipendente.

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> vif(mod=modello)
```

vif.lm()

- **Package:** `car`
- **Parametri:**

  `mod` modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Significato:** variance inflation factors
- **Formula:**

$$\left(1 - R^2_{x_j}\right)^{-1} \quad \forall\, j \,=\, 1,\, 2,\, \ldots\,,\, k-1$$

  $R^2_{x_j}$ rappresenta il valore di R^2 per il modello che presenta il regressore j-esimo come variabile dipendente.
- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> vif.lm(mod=modello)
```

outlier.test()

- **Package:** `car`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Significato:** test sugli *outliers*
- **Output:**

  `test` massimo residuo studentizzato assoluto, gradi di libertà, p-value
- **Formula:**

  `test`

$$t \,=\, \max_i(\,|\,rstudent_i\,|\,) \quad n-k-1 \quad p\text{-value} \,=\, 2\,P(\,t_{n-k-1} \,\leq\, -|\,t\,|\,) \qquad \forall\, i \,=\, 1,\, 2,\, \ldots\,,\, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> outlier.test(model=modello)
```

outlier.test.lm()

- **Package:** `car`
- **Parametri:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Significato:** test sugli *outliers*
- **Output:**

  `test` massimo residuo studentizzato assoluto, gradi di libertà, p-value
- **Formula:**

  `test`

$$t \,=\, \max_i(\,|\,rstudent_i\,|\,) \quad n-k-1 \quad p\text{-value} \,=\, 2\,P(\,t_{n-k-1} \,\leq\, -|\,t\,|\,) \qquad \forall\, i \,=\, 1,\, 2,\, \ldots\,,\, n$$

- **Esempio:**

```
> modello<-lm(formula=y~x1+x2+x3)
> outlier.test.lm(model=modello)
```

Capitolo 15

Regressione lineare multipla pesata

15.1 Simbologia

$$y_i = \beta_1 + \beta_2 x_{i1} + \beta_3 x_{i2} + \cdots + \beta_k x_{ik-1} + \varepsilon_i \quad \forall\, i = 1, 2, \ldots, n \qquad \varepsilon \sim N(0, \sigma^2\, W)$$

- variabile dipendente: y
- matrice del modello di dimensione $n \times k$: X
- numero di parametri da stimare e rango della matrice del modello: k
- numero di unità: n
- i-esima riga della matrice del modello : $X_i = (1,\, x_{i1},\, x_{i2}, \ldots,\, x_{ik-1})$
- matrice diagonale dei pesi di dimensione $n \times n$: $W = \mathrm{diag}(w_1^{-1},\, w_2^{-1},\, \ldots,\, w_n^{-1})$
- matrice di proiezione di dimensione $n \times n$: $H = X\,(X^T\, W^{-1}\, X)^{-1}\, X^T\, W^{-1}$
- matrice identità di dimensione $n \times n$: I_n
- devianza residua: $RSS = \sum_{i=1}^{n} w_i\, e_i^2 = y^T\, W^{-1}\, (I_n - H)\, y$
- stima di σ^2: $s^2 = RSS\,/\,(n-k)$
- gradi di libertà della devianza residua: $n-k$
- stima di σ^2 tolta la i-esima unità: $s^2_{-i} = s^2 \left(1 + \frac{1 - rstandard_i^2}{n-k-1}\right) \quad \forall\, i = 1, 2, \ldots, n$
- stime WLS: $\hat{\beta} = (X^T\, W^{-1}\, X)^{-1}\, X^T\, W^{-1}\, y$
- standard error delle stime WLS: $s_{\hat{\beta}} = s\,\sqrt{\mathrm{diag}((X^T\, W^{-1}\, X)^{-1})}$
- t-values delle stime WLS: $t_{\hat{\beta}} = \hat{\beta}\,/\,s_{\hat{\beta}}$
- residui: $e = (I_n - H)\, y$
- residui standard: $rstandard_i = \frac{e_i}{s\,\sqrt{(1-h_i)\,/\,w_i}} \quad \forall\, i = 1, 2, \ldots, n$
- residui studentizzati: $rstudent_i = \frac{e_i}{s_{-i}\,\sqrt{(1-h_i)\,/\,w_i}} \quad \forall\, i = 1, 2, \ldots, n$
- valori fittati: $\hat{y} = H\, y$
- valori di leva: $h = \mathrm{diag}(H)$
- stime WLS tolta la i-esima unità: $\hat{\beta}_{(-i)} \quad \forall\, i = 1, 2, \ldots, n$
- correlazione tra le stime WLS: $r_{\hat{\beta}_i\, \hat{\beta}_j} = \frac{s^2\, (X^T\, W^{-1}\, X)^{-1}_{(i,\,j)}}{s_{\hat{\beta}_i}\, s_{\hat{\beta}_j}} \quad \forall\, i,\, j = 1, 2, \ldots, k$
- devianza residua modello nullo: $RSS_{nullo} = \sum_{i=1}^{n} w_i\, (y_i - \bar{y}_W)^2 = (y - \bar{y}_W)^T\, W^{-1}\, (y - \bar{y}_W)$
- indice di determinazione: $R^2 = 1 - RSS\,/\,RSS_{nullo}$
- indice di determinazione aggiustato: $R^2_{adj} = 1 - \frac{RSS\,/\,(n-k)}{RSS_{nullo}\,/\,(n-1)}$

- valore noto dei regressori per la previsione: $x_0^T = (1, x_{01}, x_{02}, \ldots, x_{0k-1})$
- log-verosimiglianza: $\hat{\ell} = -n\left(\log(2\,\pi) + \log\left(RSS\,/\,n\right) + 1 - \sum_{i=1}^{n} \log(w_i)\,/\,n\right)\,/\,2$
- distanza di *Cook*: $cd_i = \frac{h_i\, rstandard_i^2}{k\,(1-h_i)} \quad \forall\, i = 1, 2, \ldots, n$
- covratio: $cr_i = (1-h_i)^{-1}\left(1 + \frac{rstudent_i^2 - 1}{n-k}\right)^{-k} \quad \forall i = 1, 2, \ldots, n$

15.2 Stima

lm()

- **Parametri:**

  `formula` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `weights` diagonale della matrice W^{-1}

  `x = T / F` matrice del modello

  `y = T / F` variabile dipendente

- **Significato:** analisi di regressione lineare pesata
- **Output:**

  `coefficients` stime WLS

  `residuals` residui

  `rank` rango della matrice del modello

  `fitted.values` valori fittati

  `df.residual` gradi di libertà della devianza residua

  `x` matrice del modello

  `y` variabile dipendente

- **Formula:**

  `coefficients`
  $$\hat{\beta}$$

  `residuals`
  $$e$$

  `rank`
  $$k$$

  `fitted.values`
  $$\hat{y}$$

  `df.residual`
  $$n-k$$

  `x`
  $$X$$

  `y`
  $$y$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> lm(y~x1+x2+x3,weights=w,x=T,y=T)
```

summary.lm()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità
  `correlation = T / F` matrice di correlazione delle stime WLS

- **Significato:** analisi di regressione lineare pesata

- **Output:**

  `residuals` residui
  `coefficients` stima puntuale, standard error, t-value, p-value
  `sigma` stima di σ
  `r.squared` R^2
  `adj.r.squared` R^2_{adj}
  `fstatistic` $Fvalue$, df numeratore, df denominatore
  `cov.unscaled` matrice di varianza non scalata per σ^2
  `correlation` matrice di correlazione tra le stime WLS

- **Formula:**

  `residuals`
  $$e$$

  `coefficients[,1]`
  $$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

  `coefficients[,2]`
  $$se_{\hat{\beta}_j} \quad \forall\, j = 1, 2, \ldots, k$$

  `coefficients[,3]`
  $$t_{\hat{\beta}_j} \quad \forall\, j = 1, 2, \ldots, k$$

  `coefficients[,4]`
  $$2\,P(t_{n-k} \leq -|\,t_{\hat{\beta}_j}\,|) \quad \forall\, j = 1, 2, \ldots, k$$

  `sigma`
  $$s$$

  `r.squared`
  $$R^2$$

  `adj.r.squared`
  $$R^2_{adj}$$

  `fstatistic[1]`
  $$Fvalue = \frac{(RSS_{nullo} - RSS)\,/\,(k-1)}{RSS\,/\,(n-k)}$$

  `fstatistic[2]`
  $$k-1$$

  `fstatistic[3]`
  $$n-k$$

  `cov.unscaled`
  $$(X^T\,W^{-1}\,X)^{-1}$$

  `correlation`
  $$r_{\hat{\beta}_i\,\hat{\beta}_j} \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> summary.lm(modello,correlation=T)
```

vcov()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** matrice di covarianze delle stime WLS

- **Formula:**

$$s^2 \, (X^T \, W^{-1} \, X)^{-1}$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> vcov(modello)
```

lm.wfit()

- **Parametri:**

  `X` matrice del modello

  `y` variabile dipendente

  `w` diagonale della matrice W^{-1}

- **Significato:** analisi di regressione lineare pesata

- **Output:**

  `coefficients` stime WLS

  `residuals` residui

  `fitted.values` valori fittati

  `weights` diagonale della matrice W^{-1}

  `rank` rango della matrice del modello

  `df.residual` gradi di libertà della devianza residua

- **Formula:**

  `coefficients`

$$\hat{\beta}$$

  `residuals`

$$e$$

  `fitted.values`

$$\hat{y}$$

  `weights`

$$\text{diag}(W^{-1})$$

  `rank`

$$k$$

  `df.residual`

$$n-k$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> X<-model.matrix(modello)
> lm.fit(X,y,w=w)
```

lsfit()

- **Parametri:**

  `X` matrice del modello

  `y` variabile dipendente

  `w` diagonale della matrice W^{-1}

- **Significato:** analisi di regressione lineare pesata

- **Output:**

  `coefficients` stime WLS

  `residuals` residui

  `wt` diagonale della matrice W^{-1}

- **Formula:**

  `coefficients`

  $$\hat{\beta}$$

  `residuals`

  $$e$$

  `wt`

  $$\text{diag}(W^{-1})$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> X<-model.matrix(modello)
> lsfit(X,y,w=w,intercept=F)
```

confint()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

  `parm` parametri del modello di cui vogliamo calcolare l'intervallo di confidenza

  `level` livello di confidenza $1-\alpha$

- **Significato:** intervallo di confidenza per le stime WLS

- **Formula:**

$$\hat{\beta}_j \mp t_{1-\alpha/2,\,n-k}\, s_{\hat{\beta}_j} \quad \forall j = 1, 2, \ldots, k$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> confint(modello,parm=c(1,2,3),level=0.95)
```

coef()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** stime WLS

- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> coef(modello)
```

coefficients()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** stime WLS

- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> coefficients(modello)
```

coeftest()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

  `df = NULL / Inf` significatività delle stime effettuata con la variabile casuale t oppure Z

- **Significato:** stime WLS e significatività

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> coeftest(modello,df=NULL)
```

- **Osservazioni:** E' necessario installare la libreria `lmtest`.

fitted()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** valori fittati

- **Formula:**

$$\hat{y}$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> fitted(modello)
```

fitted.values()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** valori fittati

- **Formula:**

$$\hat{y}$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> fitted.values(modello)
```

predict.lm()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

  `new` il valore di x_0

  `interval = confidence / prediction` intervallo di confidenza o previsione

  `level` livello di confidenza $1-\alpha$

  `se.fit = T / F` standard error delle stime

- **Significato:** intervallo di confidenza o di previsione

- **Output:**

  `fit` valore previsto ed intervallo di confidenza

  `se.fit` standard error delle stime

  `df` gradi di libertà della devianza residua

  `residual.scale` stima di σ

- **Formula:**

  `fit`

`interval = confidence`

$$x_0^T\,\hat{\beta} \qquad x_0^T\,\hat{\beta} \mp t_{1-\alpha\,/\,2,\,n-k}\,s\,\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

`interval = prediction`

$$x_0^T\,\hat{\beta} \qquad x_0^T\,\hat{\beta} \mp t_{1-\alpha\,/\,2,\,n-k}\,s\,\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

`se.fit`

$$s\,\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

`df`

$$n-k$$

`residual.scale`

$$s$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
```

linear.hypothesis.lm()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

  `hypothesis.matrix` matrice C di dimensione $q \times k$ e rango pari a $q = \min(q,\,k)$

  `rhs` vettore b della previsione lineare di dimensione q

- **Significato:** test di ipotesi per $H_0 : C\,\beta = b$ contro $H_1 : C\,\beta \neq b$ dove C e b sono così definiti:

$$C = \begin{pmatrix} c_{1,\,1} & c_{1,\,2} & \dots & c_{1,\,k} \\ c_{2,\,1} & c_{2,\,2} & \dots & c_{2,\,k} \\ \vdots & \vdots & \vdots & \vdots \\ c_{q,\,1} & c_{q,\,2} & \dots & c_{q,\,k} \end{pmatrix} \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_q \end{pmatrix}$$

- **Output:**

  `Res.Df` gradi di libertà della devianza residua

  `RSS` devianza residua

  `Df` gradi di libertà della devianza relativa all'ipotesi nulla H_0

  `Sum of Sq` devianza relativa all'ipotesi nulla H_0

  `F` valore empirico della statistica F

  `Pr(>F)` p-value

- **Formula:**

  `Res.Df`

$$n-k \qquad n-k+q$$

  `RSS`

$$RSS \qquad RSS + \left(b - C\,\hat{\beta}\right)^T \left[C\,\left(X^T\,W^{-1}\,X\right)^{-1}\,C^T\right]^{-1}\,\left(b - C\,\hat{\beta}\right)$$

  `Df`

$$-q$$

  `Sum of Sq`

$$-\left(b - C\,\hat{\beta}\right)^T \left[C\,\left(X^T\,W^{-1}\,X\right)^{-1}\,C^T\right]^{-1}\,\left(b - C\,\hat{\beta}\right)$$

  `F`

$$Fvalue = \frac{\left[\left(b - C\,\hat{\beta}\right)^T \left[C\,\left(X^T\,W^{-1}\,X\right)^{-1}\,C^T\right]^{-1}\,\left(b - C\,\hat{\beta}\right)\right] / q}{RSS\,/\,(n-k)}$$

`Pr(>F)`

$$P(F_{q,\,n-k} \geq Fvalue)$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> k<-4
> q<-2
> C
     [,1] [,2] [,3] [,4]
[1,]    1    3  5.0  2.3
[2,]    2    4  1.1  4.3
> b
     [,1]
[1,]  1.1
[2,]  2.3
> linear.hypothesis.lm(modello,hypothesis.matrix=C,rhs=b)
```

- **Osservazioni:** E' necessario installare la libreria `car`.

15.3 Adattamento

durbin.watson()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui

- **Output:**

  `dw` valore empirico della statistica D–W

- **Formula:**

  `dw`

$$\frac{\sum_{i=2}^{n}(e_i - e_{i-1})^2}{RSS}$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> durbin.watson(modello)
```

- **Osservazioni:** E' necessario installare la libreria `car`.

logLik()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** log-verosimiglianza normale pesata

- **Formula:**

$$\hat{\ell}(\mu,\, \sigma^2)$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> logLik(modello)
```

deviance()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** devianza residua

- **Formula:**

$$RSS$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> deviance(modello)
```

AIC()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** indice AIC

- **Formula:**

$$-2\,\hat{\ell} + 2\,(k+1)$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> AIC(modello)
```

BIC()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** indice BIC

- **Formula:**

$$-2\,\hat{\ell} + (k+1)\,\log(n)$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> BIC(modello)
```

- **Osservazioni:** E' necessario installare la libreria `nlme`.

extractAIC()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** numero di parametri del modello ed indice AIC generalizzato

- **Formula:**

$$k \quad n \log(RSS \,/\, n) + 2\,k$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> extractAIC(modello)
```

15.4 Diagnostica

ls.diag()

- **Parametri:**

  `formula` oggetto di tipo `lsfit()`

- **Significato:** analisi di regressione lineare pesata

- **Output:**

  `std.dev` stima di σ

  `hat` valori di leva

  `std.res` residui standard

  `stud.res` residui studentizzati

  `cooks` distanza di *Cook*

  `dfits` dfits

  `correlation` matrice di correlazione tra le stime WLS

  `std.err` standard error delle stime WLS

  `cov.scaled` matrice di covarianza delle stime WLS

  `cov.unscaled` matrice di covarianza delle stime WLS non scalata per σ^2

- **Formula:**

  `std.dev`

$$s$$

  `hat`

$$h$$

  `std.res`

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

  `stud.res`

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

  `cooks`

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

  `dfits`

$$rstudent_i \sqrt{\frac{h_i}{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$$

  `correlation`

$$r_{\hat{\beta}_i\,\hat{\beta}_j} \quad \forall\, i, j = 1, 2, \ldots, k$$

`std.err`

$$s_{\hat{\beta}_j} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k$$

`cov.scaled`

$$s^2\,(X^T\,W^{-1}\,X)^{-1}$$

`cov.unscaled`

$$(X^T\,W^{-1}\,X)^{-1}$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lsfit(X,y,w=w,intercept=F)
> ls.diag(modello)
```

lm.influence()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** diagnostica di regressione

- **Output:**

  `hat` valori di leva

  `coefficients` differenza tra le stime WLS eliminando una unità

  `sigma` stima di σ eliminando una unità

  `wt.res` residui pesati

- **Formula:**

  `hat`

  $$h$$

  `coefficients`

  $$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} \,=\, w_i\,e_i\,(1-h_i)^{-1}\,(X^T\,W^{-1}\,X)^{-1}_j\,X^T_i \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n \quad \forall j \,=\, 1,\, 2,\, \ldots,\, k$$

  `sigma`

  $$s_{-i} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

  `wt.res`

  $$\sqrt{w_i}\,e_i \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> lm.influence(modello)
```

influence()

- **Parametri:**

  `formula` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** diagnostica di regressione

- **Output:**

  `hat` valori di leva

  `coefficients` differenza tra le stime WLS eliminando una unità

  `sigma` stima di σ eliminando una unità

  `wt.res` residui pesati

- **Formula:**

  `hat`

  $$h$$

  `coefficients`

  $$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = w_i\, e_i\, (1-h_i)^{-1}\, (X^T\, W^{-1}\, X)^{-1}_j\, X^T_i \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

  `sigma`

  $$s_{-i} \quad \forall\, i = 1, 2, \ldots, n$$

  `wt.res`

  $$\sqrt{w_i}\, e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> influence(modello)
```

weights()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** pesi

- **Formula:**

  $$\text{diag}(W^{-1})$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> weights(modello)
```

weighted.residuals()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** residui pesati

- **Formula:**

$$\sqrt{w_i}\, e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> weighted.residuals(modello)
```

residuals()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

  `type = response / pearson` tipo di residuo

- **Significato:** residui

- **Formula:**

$$\boxed{\texttt{type = response}}$$

$$e$$

$$\boxed{\texttt{type = pearson}}$$

$$\sqrt{w_i}\, e_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> residuals(modello,type="response")
```

residuals.default()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** residui

- **Formula:**

$$e$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> residuals.default(modello)
```

resid()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** residui

- **Formula:**

$$e$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> resid(modello)
```

outlier.test()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** test sugli *outliers*

- **Output:**

  `test` verifica di ipotesi

- **Formula:**

  `test[1]`

$$t = \max\left(\left\{ \left| rstudent_i \right| \right\}_{i=1,2,\dots,n} \right)$$

  `test[2]`

$$n-k-1$$

  `test[3]`

$$p = 2\,P(t_{n-k-1} \leq -|\,t\,|)$$

  `test[4]`

$$\begin{cases} np & \text{se } np \leq 1 \\ \texttt{NA} & \text{se } np > 1 \end{cases}$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> outlier.test.lm(modello)
```

- **Osservazioni:** E' necessario installare la libreria `car`.

df.residual()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** gradi di libertà della devianza residua

- **Formula:**

$$n-k$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> df.residual(modello)
```

hatvalues()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** valori di leva

- **Formula:**

$$h$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> hatvalues(modello)
```

hat()

- **Parametri:**

  `X` matrice del modello

- **Significato:** valori di leva

- **Formula:**

$$h$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> X<-model.matrix(modello)
> hat(X,intercept=T)
```

rstandard()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** residui standard

- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> rstandard(modello)
```

stdres()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** residui standard

- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> stdres(modello)
```

- **Osservazioni:** E' necessario installare la libreria `MASS`.

rstudent()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** residui studentizzati

- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> rstudent(modello)
```

studres()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** residui studentizzati

- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> studres(modello)
```

- **Osservazioni:** E' necessario installare la libreria `MASS`.

lmwork()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** diagnostica di regressione

- **Output:**

  `stdedv` stima di σ

  `stdres` residui standard

  `studres` residui studentizzati

- **Formula:**

  `stdedv`

$$s$$

  `stdres`

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

  `studres`

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> lmwork(modello)
```

- **Osservazioni:** E' necessario installare la libreria `MASS`.

dffits()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** dffits

- **Formula:**

$$rstudent_i \sqrt{\frac{h_i}{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> dffits(modello)
```

covratio()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** covratio

- **Formula:**

$$cr_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> covratio(modello)
```

cooks.distance()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** distanza di *Cook*

- **Formula:**

$$cd_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> cooks.distance(modello)
```

cookd()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** distanza di *Cook*

- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> cookd(modello)
```

- **Osservazioni:** E' necessario installare la libreria `car`.

dfbeta()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** dfbeta

- **Formula:**

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = w_i\, e_i\, (1-h_i)^{-1}\, (X^T\, W^{-1}\, X)^{-1}_j\, X^T_i \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> dfbeta(modello)
```

dfbetas()

- **Parametri:**

  `formula` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Significato:** dfbetas

- **Formula:**

$$\frac{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}{s_{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}} = \frac{w_i\, e_i\, (1-h_i)^{-1}\, (X^T\, W^{-1}\, X)^{-1}_j\, X^T_i}{s_{-i}\, \sqrt{(X^T\, W^{-1}\, X)^{-1}_{j,\,j}}} \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> dfbetas(modello)
```

vif()

- **Parametri:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Significato:** variance inflation factors

- **Formula:**

$$\left(1 - R^2_{x_j}\right)^{-1} \quad \forall\, j \,=\, 1,\, 2,\, \ldots\,,\, k-1$$

  $R^2_{x_j}$ rappresenta il valore di R^2 per il modello che presenta il regressore j-esimo come variabile dipendente.

- **Esempio:**

```
> n<-length(y)
> w<-abs(rnorm(n))
> modello<-lm(formula=y~x1+x2+x3,weights=w)
> vif.lm(modello)
```

- **Osservazioni:** E' necessario installare la libreria `car`.

Parte V

Modelli Lineari Generalizzati

Capitolo 16

Regressione Logit

16.1 Simbologia

$$\log\left(\frac{\pi_i}{1-\pi_i}\right) = \beta_1 + \beta_2 x_{i1} + \beta_3 x_{i2} + \cdots + \beta_k x_{ik-1} \qquad Y_i \sim \text{Bin}(\pi_i, n_i) \quad \forall\, i = 1, 2, \ldots, n$$

- numero di successi: $y_i \quad \forall\, i = 1, 2, \ldots, n$
- numero di prove: $n_i \quad \forall\, i = 1, 2, \ldots, n$
- matrice del modello di dimensione $n \times k$: X
- numero di parametri da stimare e rango della matrice del modello: k
- numero di unità: n
- i-esima riga della matrice del modello : $X_i = (1, x_{i1}, x_{i2}, \ldots, x_{ik-1})$
- matrice diagonale dei pesi IWLS di dimensione $n \times n$: $W = \text{diag}(w_1^{-1}, w_2^{-1}, \ldots, w_n^{-1})$
- matrice di proiezione di dimensione $n \times n$: $H = X\,(X^T\,W^{-1}\,X)^{-1}\,X^T\,W^{-1}$
- devianza residua:
  $D = 2\sum_{i=1}^{n}\left[y_i \log\left(\frac{y_i}{\hat{y}_i} + \frac{1}{2}\,\frac{1-\text{sign}(y_i)}{\hat{y}_i}\right) + (n_i - y_i)\log\left(\frac{n_i-y_i}{n_i-\hat{y}_i} + \frac{1}{2}\,\frac{1-\text{sign}(n_i-y_i)}{n_i-\hat{y}_i}\right)\right] = \sum_{i=1}^{n} e_i^2$
  dove $\hat{y}_i = n_i\,\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$
- gradi di libertà della devianza residua: $n - k$
- stime: $\hat{\beta}$
- standard error delle stime: $s_{\hat{\beta}} = \sqrt{\text{diag}((X^T\,W^{-1}\,X)^{-1})}$
- z-values delle stime: $z_{\hat{\beta}} = \hat{\beta}\,/\,s_{\hat{\beta}}$
- residui: $e_i \quad \forall\, i = 1, 2, \ldots, n$
- residui standard: $rstandard_i = \frac{e_i}{\sqrt{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$
- valori fittati: $\hat{\pi}_i = \frac{\exp\left(X_i\,\hat{\beta}\right)}{1+\exp\left(X_i\,\hat{\beta}\right)} \quad \forall\, i = 1, 2, \ldots, n$
- valori di leva: $h = \text{diag}(H)$
- devianza residua modello nullo:
  $D_{nullo} = 2\sum_{i=1}^{n}\left[y_i \log\left(\frac{y_i}{\hat{y}_i} + \frac{1}{2}\,\frac{1-\text{sign}(y_i)}{\hat{y}_i}\right) + (n_i - y_i)\log\left(\frac{n_i-y_i}{n_i-\hat{y}_i} + \frac{1}{2}\,\frac{1-\text{sign}(n_i-y_i)}{n_i-\hat{y}_i}\right)\right]$
  dove $\hat{y}_i = n_i\,\hat{\pi} = n_i\,\frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} n_i} \quad \forall\, i = 1, 2, \ldots, n$
- log-verosimiglianza binomiale: $\hat{\ell} = \sum_{i=1}^{n}\left[\log\binom{n_i}{y_i} + y_i \log\left(\frac{\hat{y}_i}{n_i}\right) + (n_i - y_i)\log\left(1 - \frac{\hat{y}_i}{n_i}\right)\right]$
  dove $\hat{y}_i = n_i\,\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$

16.2 Stima

glm()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

  `x = T / F` matrice del modello

- **Significato:** analisi di regressione logit
- **Output:**

  `coefficients` stime

  `fitted.values` valori fittati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` della devianza residua

  `aic` indice AIC

  `null.deviance` devianza residua modello nullo

  `weights` pesi IWLS

  `prior.weights` pesi iniziali

  `df.residual` gradi di libertà devianza residua

  `df.null` gradi di libertà devianza residua modello nullo

  `y` proporzione di successi

  `x` matrice del modello

- **Formula:**

  `coefficients`

  $$\hat{\beta}$$

  `fitted.values`

  $$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

  `rank`

  $$k$$

  `linear.predictors`

  $$X\,\hat{\beta}$$

  `deviance`

  $$D$$

  `aic`

  $$-2\,\hat{\ell} + 2\,k$$

  `null.deviance`

  $$D_{nullo}$$

  `weights`

  $$w_i \quad \forall\, i = 1, 2, \ldots, n$$

  `prior.weights`

  $$n_i \quad \forall\, i = 1, 2, \ldots, n$$

  `df.residual`

  $$n - k$$

  `df.null`

  $$n - 1$$

  `y`

  $$y_i \,/\, n_i \quad \forall\, i = 1, 2, \ldots, n$$

`x`

$$X$$

- **Esempio:**

```
> glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"),x=T)
```

summary.glm()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** analisi di regressione logit
- **Output:**

  `deviance` devianza residua

  `aic` indice AIC

  `df.residual` gradi di libertà devianza residua

  `null.deviance` devianza residua modello nullo

  `df.null` gradi di libertà devianza residua modello nullo

  `deviance.resid` residui di devianza

  `coefficients` stima puntuale, standard error, z-value, p-value

  `cov.unscaled` matrice di covarianza delle stime non scalata

  `cov.scaled` matrice di covarianza delle stime scalata

- **Formula:**

  `deviance`

  $$D$$

  `aic`

  $$-2\,\hat{\ell} + 2\,k$$

  `df.residual`

  $$n - k$$

  `df.null`

  $$n - 1$$

  `deviance.resid`

  $$e$$

  `coefficients`

  $$\hat{\beta}_j \quad s_{\hat{\beta}_j} \quad z_{\hat{\beta}_j} \quad p\text{-value} = 2\,\Phi(-\,|\,z_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1, 2, \ldots, k$$

  `cov.unscaled`

  $$(X^T\,W^{-1}\,X)^{-1}$$

  `cov.scaled`

  $$(X^T\,W^{-1}\,X)^{-1}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> summary.glm(modello)
```

glm.fit()

- **Package:** `stats`
- **Parametri:**

  `x` matrice del modello
  `y` proporzione di successi
  `weights` numero di prove

- **Significato:** analisi di regressione logit
- **Output:**

  `coefficients` stime
  `fitted.values` valori fittati
  `rank` rango della matrice del modello
  `linear.predictors` predittori lineari
  `deviance` della devianza residua
  `aic` indice AIC
  `null.deviance` devianza residua modello nullo
  `weights` pesi IWLS
  `prior.weights` pesi iniziali
  `df.residual` gradi di libertà devianza residua
  `df.null` gradi di libertà devianza residua modello nullo
  `y` proporzione di successi

- **Formula:**

  `coefficients`
  $$\hat{\beta}$$

  `fitted.values`
  $$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

  `rank`
  $$k$$

  `linear.predictors`
  $$X\,\hat{\beta}$$

  `deviance`
  $$D$$

  `aic`
  $$-2\,\hat{\ell} + 2\,k$$

  `null.deviance`
  $$D_{nullo}$$

  `weights`
  $$w_i \quad \forall\, i = 1, 2, \ldots, n$$

  `prior.weights`
  $$n_i \quad \forall\, i = 1, 2, \ldots, n$$

  `df.residual`
  $$n - k$$

  `df.null`
  $$n - 1$$

  `y`
  $$y_i \,/\, n_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> glm.fit(X,y/Total,weights=Total,family=binomial(link="logit"))
```

vcov()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** matrice di covarianza delle stime
- **Formula:**

$$s^2 \, (X^T \, W^{-1} \, X)^{-1}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> vcov(modello)
```

coef()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** stime
- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> coef(modello)
```

coefficients()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** stime
- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> coefficients(modello)
```

fitted()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** valori fittati
- **Formula:**

$$\hat{y}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> fitted(modello)
```

fitted.values()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** valori fittati
- **Formula:**

$$\hat{y}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> fitted.values(modello)
```

16.3 Adattamento

logLik()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** log-verosimiglianza normale
- **Formula:**

$$\hat{\ell}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> logLik(modello)
```

AIC()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** indice AIC
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> AIC(modello)
```

extractAIC()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** indice AIC generalizzato
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> extractAIC(modello)
```

deviance()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** devianza residua
- **Formula:**

$$D$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> deviance(modello)
```

anova()

- **Package:** `stats`
- **Parametri:**

  `nullo` modello di regressione logit nullo con n unità

  `formula` modello di regressione logit con $k-1$ variabili esplicative con n unità

- **Significato:** anova di regressione
- **Output:**

  `Resid. Df` gradi di libertà

  `Resid. Dev` devianza residua

  `Df` differenza dei gradi di libertà

  `Deviance` differenza tra le devianze residue

  `P(>|Chi|)` p-value

- **Formula:**

  `Resid. Df`

  $$n-1 \quad n-k$$

  `Resid. Dev`

  $$D_{nullo} \quad D$$

  `Df`

  $$1$$

  `Deviance`

  $$c = D_{nullo} - D$$

  `P(>|Chi|)`

  $$P(\chi^2_{k-1} \geq c)$$

- **Esempio:**

16.4 Diagnostica

rstandard()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> rstandard(modello)
```

residuals()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

  `type = deviance / pearson` tipo di residuo

- **Significato:** residui
- **Formula:**

$$\boxed{\texttt{type = deviance}}$$

$$e$$

$$\boxed{\texttt{type = pearson}}$$

$$\frac{y_i - n_i\,\hat{\pi}_i}{\sqrt{n_i\,\hat{\pi}_i\,(1-\hat{\pi}_i)}} \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> residuals(modello,type="deviance")
```

resid()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

  `type = deviance / pearson` tipo di residuo

- **Significato:** residui
- **Formula:**

$$\boxed{\texttt{type = deviance}}$$

$$e$$

$$\boxed{\texttt{type = pearson}}$$

$$\frac{y_i - n_i\,\hat{\pi}_i}{\sqrt{n_i\,\hat{\pi}_i\,(1-\hat{\pi}_i)}} \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> resid(modello,type="deviance")
```

df.residual()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** gradi di libertà della devianza residua
- **Formula:**

$$n-k$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> df.residual(modello)
```

hatvalues()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Significato:** valori di leva
- **Formula:**

$$h$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="logit"))
> hatvalues(modello)
```

Capitolo 17

Regressione Probit

17.1 Simbologia

$$\Phi^{-1}(\pi_i) = \beta_1 + \beta_2 x_{i1} + \beta_3 x_{i2} + \cdots + \beta_k x_{ik-1} \qquad Y_i \sim \text{Bin}(\pi_i, n_i) \quad \forall\, i = 1, 2, \ldots, n$$

- numero di successi: $y_i \quad \forall\, i = 1, 2, \ldots, n$
- numero di prove: $n_i \quad \forall\, i = 1, 2, \ldots, n$
- matrice del modello di dimensione $n \times k$: X
- numero di parametri da stimare e rango della matrice del modello: k
- numero di unità: n
- i-esima riga della matrice del modello : $X_i = (1, x_{i1}, x_{i2}, \ldots, x_{ik-1})$
- matrice diagonale dei pesi IWLS di dimensione $n \times n$: $W = \text{diag}(w_1^{-1}, w_2^{-1}, \ldots, w_n^{-1})$
- matrice di proiezione di dimensione $n \times n$: $H = X\,(X^T\,W^{-1}\,X)^{-1}\,X^T\,W^{-1}$
- devianza residua:
  $D = 2\sum_{i=1}^{n}\left[y_i \log\left(\frac{y_i}{\hat{y}_i} + \frac{1}{2}\,\frac{1-\text{sign}(y_i)}{\hat{y}_i}\right) + (n_i - y_i)\log\left(\frac{n_i-y_i}{n_i-\hat{y}_i} + \frac{1}{2}\,\frac{1-\text{sign}(n_i-y_i)}{n_i-\hat{y}_i}\right)\right] = \sum_{i=1}^{n} e_i^2$
  dove $\hat{y}_i = n_i\,\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$
- gradi di libertà della devianza residua: $n - k$
- stime: $\hat{\beta}$
- standard error delle stime: $s_{\hat{\beta}} = \sqrt{\text{diag}((X^T\,W^{-1}\,X)^{-1})}$
- z-values delle stime: $z_{\hat{\beta}} = \hat{\beta}\,/\,s_{\hat{\beta}}$
- residui: $e_i \quad \forall\, i = 1, 2, \ldots, n$
- residui standard: $rstandard_i = \frac{e_i}{\sqrt{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$
- valori fittati: $\hat{\pi}_i = \Phi\left(X_i\,\hat{\beta}\right) \quad \forall\, i = 1, 2, \ldots, n$
- valori di leva: $h = \text{diag}(H)$
- devianza residua modello nullo:
  $D_{nullo} = 2\sum_{i=1}^{n}\left[y_i \log\left(\frac{y_i}{\hat{y}_i} + \frac{1}{2}\,\frac{1-\text{sign}(y_i)}{\hat{y}_i}\right) + (n_i - y_i)\log\left(\frac{n_i-y_i}{n_i-\hat{y}_i} + \frac{1}{2}\,\frac{1-\text{sign}(n_i-y_i)}{n_i-\hat{y}_i}\right)\right]$
  dove $\hat{y}_i = n_i\,\hat{\pi} = n_i\,\frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} n_i} \quad \forall\, i = 1, 2, \ldots, n$
- log-verosimiglianza binomiale: $\hat{\ell} = \sum_{i=1}^{n}\left[\log\binom{n_i}{y_i} + y_i \log\left(\frac{\hat{y}_i}{n_i}\right) + (n_i - y_i)\log\left(1 - \frac{\hat{y}_i}{n_i}\right)\right]$
  dove $\hat{y}_i = n_i\,\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$

17.2 Stima

glm()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

  `x = T / F` matrice del modello

- **Significato:** analisi di regressione probit
- **Output:**

  `coefficients` stime

  `fitted.values` valori fittati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` della devianza residua

  `aic` indice AIC

  `null.deviance` devianza residua modello nullo

  `weights` pesi IWLS

  `prior.weights` pesi iniziali

  `df.residual` gradi di libertà devianza residua

  `df.null` gradi di libertà devianza residua modello nullo

  `y` proporzione di successi

  `x` matrice del modello

- **Formula:**

  `coefficients`

  $$\hat{\beta}$$

  `fitted.values`

  $$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

  `rank`

  $$k$$

  `linear.predictors`

  $$X\,\hat{\beta}$$

  `deviance`

  $$D$$

  `aic`

  $$-2\,\hat{\ell} + 2\,k$$

  `null.deviance`

  $$D_{nullo}$$

  `weights`

  $$w_i \quad \forall\, i = 1, 2, \ldots, n$$

  `prior.weights`

  $$n_i \quad \forall\, i = 1, 2, \ldots, n$$

  `df.residual`

  $$n - k$$

  `df.null`

  $$n - 1$$

  `y`

  $$y_i \,/\, n_i \quad \forall\, i = 1, 2, \ldots, n$$

`x`

$$X$$

- **Esempio:**

```
> glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"),x=T)
```

summary.glm()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** analisi di regressione probit
- **Output:**

  `deviance` devianza residua

  `aic` indice AIC

  `df.residual` gradi di libertà devianza residua

  `null.deviance` devianza residua modello nullo

  `df.null` gradi di libertà devianza residua modello nullo

  `deviance.resid` residui di devianza

  `coefficients` stima puntuale, standard error, z-value, p-value

  `cov.unscaled` matrice di covarianza delle stime non scalata

  `cov.scaled` matrice di covarianza delle stime scalata

- **Formula:**

  `deviance`

  $$D$$

  `aic`

  $$-2\,\hat{\ell} + 2\,k$$

  `df.residual`

  $$n - k$$

  `df.null`

  $$n - 1$$

  `deviance.resid`

  $$e$$

  `coefficients`

  $$\hat{\beta}_j \quad s_{\hat{\beta}_j} \quad z_{\hat{\beta}_j} \quad p\text{-value} = 2\,\Phi(-\,|\,z_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1, 2, \ldots, k$$

  `cov.unscaled`

  $$(X^T\,W^{-1}\,X)^{-1}$$

  `cov.scaled`

  $$(X^T\,W^{-1}\,X)^{-1}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> summary.glm(modello)
```

glm.fit()

- **Package:** `stats`
- **Parametri:**

  `x` matrice del modello
  `y` proporzione di successi
  `weights` numero di prove

- **Significato:** analisi di regressione probit
- **Output:**

  `coefficients` stime
  `fitted.values` valori fittati
  `rank` rango della matrice del modello
  `linear.predictors` predittori lineari
  `deviance` della devianza residua
  `aic` indice AIC
  `null.deviance` devianza residua modello nullo
  `weights` pesi IWLS
  `prior.weights` pesi iniziali
  `df.residual` gradi di libertà devianza residua
  `df.null` gradi di libertà devianza residua modello nullo
  `y` proporzione di successi

- **Formula:**

  `coefficients`
  $$\hat{\beta}$$

  `fitted.values`
  $$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

  `rank`
  $$k$$

  `linear.predictors`
  $$X\,\hat{\beta}$$

  `deviance`
  $$D$$

  `aic`
  $$-2\,\hat{\ell} + 2\,k$$

  `null.deviance`
  $$D_{nullo}$$

  `weights`
  $$w_i \quad \forall\, i = 1, 2, \ldots, n$$

  `prior.weights`
  $$n_i \quad \forall\, i = 1, 2, \ldots, n$$

  `df.residual`
  $$n - k$$

  `df.null`
  $$n - 1$$

  `y`
  $$y_i \,/\, n_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> glm.fit(X,y/Total,weights=Total,family=binomial(link="probit"))
```

vcov()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** matrice di covarianza delle stime
- **Formula:**

$$s^2 \, (X^T \, W^{-1} \, X)^{-1}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> vcov(modello)
```

coef()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** stime
- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> coef(modello)
```

coefficients()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** stime
- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> coefficients(modello)
```

fitted()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** valori fittati
- **Formula:**

$$\hat{y}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> fitted(modello)
```

fitted.values()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** valori fittati
- **Formula:**

$$\hat{y}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> fitted.values(modello)
```

17.3 Adattamento

logLik()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** log-verosimiglianza normale
- **Formula:**

$$\hat{\ell}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> logLik(modello)
```

AIC()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** indice AIC
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> AIC(modello)
```

extractAIC()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** indice AIC generalizzato
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> extractAIC(modello)
```

deviance()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** devianza residua
- **Formula:**

$$D$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> deviance(modello)
```

anova()

- **Package:** `stats`
- **Parametri:**

  `nullo` modello di regressione probit nullo con n unità

  `formula` modello di regressione probit con $k-1$ variabili esplicative con n unità

- **Significato:** anova di regressione
- **Output:**

  `Resid. Df` gradi di libertà

  `Resid. Dev` devianza residua

  `Df` differenza dei gradi di libertà

  `Deviance` differenza tra le devianze residue

  `P(>|Chi|)` p-value

- **Formula:**

  `Resid. Df`

  $$n-1 \quad n-k$$

  `Resid. Dev`

  $$D_{nullo} \quad D$$

  `Df`

  $$1$$

  `Deviance`

  $$c = D_{nullo} - D$$

  `P(>|Chi|)`

  $$P(\chi^2_{k-1} \geq c)$$

- **Esempio:**

17.4 Diagnostica

rstandard()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> rstandard(modello)
```

residuals()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

  `type = deviance / pearson` tipo di residuo

- **Significato:** residui
- **Formula:**

$$\boxed{\texttt{type = deviance}}$$

$$e$$

$$\boxed{\texttt{type = pearson}}$$

$$\frac{y_i - n_i\,\hat{\pi}_i}{\sqrt{n_i\,\hat{\pi}_i\,(1-\hat{\pi}_i)}} \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> residuals(modello,type="deviance")
```

resid()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

  `type = deviance / pearson` tipo di residuo

- **Significato:** residui
- **Formula:**

$$\boxed{\texttt{type = deviance}}$$

$$e$$

$$\boxed{\texttt{type = pearson}}$$

$$\frac{y_i - n_i\,\hat{\pi}_i}{\sqrt{n_i\,\hat{\pi}_i\,(1-\hat{\pi}_i)}} \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> resid(modello,type="deviance")
```

df.residual()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** gradi di libertà della devianza residua
- **Formula:**

$$n-k$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> df.residual(modello)
```

hatvalues()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Significato:** valori di leva
- **Formula:**

$$h$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="probit"))
> hatvalues(modello)
```

Capitolo 18

Regressione Complementary log-log

18.1 Simbologia

$$\log\left(-\log\left(1-\pi_i\right)\right) = \beta_1 + \beta_2 x_{i1} + \beta_3 x_{i2} + \cdots + \beta_k x_{ik-1} \qquad Y_i \sim \text{Bin}(\pi_i, n_i) \quad \forall\, i = 1, 2, \ldots, n$$

- numero di successi: $y_i \quad \forall\, i = 1, 2, \ldots, n$
- numero di prove: $n_i \quad \forall\, i = 1, 2, \ldots, n$
- matrice del modello di dimensione $n \times k$: X
- numero di parametri da stimare e rango della matrice del modello: k
- numero di unità: n
- i-esima riga della matrice del modello : $X_i = (1, x_{i1}, x_{i2}, \ldots, x_{ik-1})$
- matrice diagonale dei pesi IWLS di dimensione $n \times n$: $W = \text{diag}(w_1^{-1}, w_2^{-1}, \ldots, w_n^{-1})$
- matrice di proiezione di dimensione $n \times n$: $H = X\,(X^T\,W^{-1}\,X)^{-1}\,X^T\,W^{-1}$
- devianza residua:
  $D = 2\sum_{i=1}^{n}\left[y_i\,\log\left(\frac{y_i}{\hat{y}_i} + \frac{1}{2}\,\frac{1-\text{sign}(y_i)}{\hat{y}_i}\right) + (n_i - y_i)\,\log\left(\frac{n_i-y_i}{n_i-\hat{y}_i} + \frac{1}{2}\,\frac{1-\text{sign}(n_i-y_i)}{n_i-\hat{y}_i}\right)\right] = \sum_{i=1}^{n} e_i^2$
  dove $\hat{y}_i = n_i\,\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$
- gradi di libertà della devianza residua: $n - k$
- stime: $\hat{\beta}$
- standard error delle stime: $s_{\hat{\beta}} = \sqrt{\text{diag}((X^T\,W^{-1}\,X)^{-1})}$
- z-values delle stime: $z_{\hat{\beta}} = \hat{\beta}\,/\,s_{\hat{\beta}}$
- residui: $e_i \quad \forall\, i = 1, 2, \ldots, n$
- residui standard: $rstandard_i = \frac{e_i}{\sqrt{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$
- valori fittati: $\hat{\pi}_i = 1 - \exp\left(-\exp\left(X_i\,\hat{\beta}\right)\right) \quad \forall\, i = 1, 2, \ldots, n$
- valori di leva: $h = \text{diag}(H)$
- devianza residua modello nullo:
  $D_{nullo} = 2\sum_{i=1}^{n}\left[y_i\,\log\left(\frac{y_i}{\hat{y}_i} + \frac{1}{2}\,\frac{1-\text{sign}(y_i)}{\hat{y}_i}\right) + (n_i - y_i)\,\log\left(\frac{n_i-y_i}{n_i-\hat{y}_i} + \frac{1}{2}\,\frac{1-\text{sign}(n_i-y_i)}{n_i-\hat{y}_i}\right)\right]$
  dove $\hat{y}_i = n_i\,\hat{\pi} = n_i\,\frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} n_i} \quad \forall\, i = 1, 2, \ldots, n$
- log-verosimiglianza binomiale: $\hat{\ell} = \sum_{i=1}^{n}\left[\log\binom{n_i}{y_i} + y_i\,\log\left(\frac{\hat{y}_i}{n_i}\right) + (n_i - y_i)\,\log\left(1 - \frac{\hat{y}_i}{n_i}\right)\right]$
  dove $\hat{y}_i = n_i\,\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$

18.2 Stima

glm()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità
  `x = T / F` matrice del modello

- **Significato:** analisi di regressione log-log complementare
- **Output:**

  `coefficients` stime
  `fitted.values` valori fittati
  `rank` rango della matrice del modello
  `linear.predictors` predittori lineari
  `deviance` della devianza residua
  `aic` indice AIC
  `null.deviance` devianza residua modello nullo
  `weights` pesi IWLS
  `prior.weights` pesi iniziali
  `df.residual` gradi di libertà devianza residua
  `df.null` gradi di libertà devianza residua modello nullo
  `y` proporzione di successi
  `x` matrice del modello

- **Formula:**

  `coefficients`
  $$\hat{\beta}$$

  `fitted.values`
  $$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

  `rank`
  $$k$$

  `linear.predictors`
  $$X\,\hat{\beta}$$

  `deviance`
  $$D$$

  `aic`
  $$-2\,\hat{\ell} + 2\,k$$

  `null.deviance`
  $$D_{nullo}$$

  `weights`
  $$w_i \quad \forall\, i = 1, 2, \ldots, n$$

  `prior.weights`
  $$n_i \quad \forall\, i = 1, 2, \ldots, n$$

  `df.residual`
  $$n - k$$

  `df.null`
  $$n - 1$$

  `y`
  $$y_i \,/\, n_i \quad \forall\, i = 1, 2, \ldots, n$$

`x`

$$X$$

- **Esempio:**

```
> glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"),x=T)
```

summary.glm()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** analisi di regressione log-log complementare
- **Output:**

  `deviance` devianza residua

  `aic` indice AIC

  `df.residual` gradi di libertà devianza residua

  `null.deviance` devianza residua modello nullo

  `df.null` gradi di libertà devianza residua modello nullo

  `deviance.resid` residui di devianza

  `coefficients` stima puntuale, standard error, z-value, p-value

  `cov.unscaled` matrice di covarianza delle stime non scalata

  `cov.scaled` matrice di covarianza delle stime scalata

- **Formula:**

  `deviance`

  $$D$$

  `aic`

  $$-2\,\hat{\ell} + 2\,k$$

  `df.residual`

  $$n - k$$

  `df.null`

  $$n - 1$$

  `deviance.resid`

  $$e$$

  `coefficients`

  $$\hat{\beta}_j \quad s_{\hat{\beta}_j} \quad z_{\hat{\beta}_j} \quad p\text{-value} = 2\,\Phi(-\,|\,z_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1, 2, \ldots, k$$

  `cov.unscaled`

  $$(X^T\,W^{-1}\,X)^{-1}$$

  `cov.scaled`

  $$(X^T\,W^{-1}\,X)^{-1}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> summary.glm(modello)
```

glm.fit()

- **Package:** `stats`
- **Parametri:**

  `x` matrice del modello
  `y` proporzione di successi
  `weights` numero di prove

- **Significato:** analisi di regressione log-log complementare
- **Output:**

  `coefficients` stime
  `fitted.values` valori fittati
  `rank` rango della matrice del modello
  `linear.predictors` predittori lineari
  `deviance` della devianza residua
  `aic` indice AIC
  `null.deviance` devianza residua modello nullo
  `weights` pesi IWLS
  `prior.weights` pesi iniziali
  `df.residual` gradi di libertà devianza residua
  `df.null` gradi di libertà devianza residua modello nullo
  `y` proporzione di successi

- **Formula:**

  `coefficients`
  $$\hat{\beta}$$

  `fitted.values`
  $$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

  `rank`
  $$k$$

  `linear.predictors`
  $$X\,\hat{\beta}$$

  `deviance`
  $$D$$

  `aic`
  $$-2\,\hat{\ell} + 2\,k$$

  `null.deviance`
  $$D_{nullo}$$

  `weights`
  $$w_i \quad \forall\, i = 1, 2, \ldots, n$$

  `prior.weights`
  $$n_i \quad \forall\, i = 1, 2, \ldots, n$$

  `df.residual`
  $$n - k$$

  `df.null`
  $$n - 1$$

  `y`
  $$y_i \,/\, n_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> glm.fit(X,y/Total,weights=Total,family=binomial(link="cloglog"))
```

vcov()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** matrice di covarianza delle stime
- **Formula:**

$$s^2 \, (X^T \, W^{-1} \, X)^{-1}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> vcov(modello)
```

coef()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** stime
- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> coef(modello)
```

coefficients()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** stime
- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> coefficients(modello)
```

fitted()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** valori fittati
- **Formula:**

$$\hat{y}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> fitted(modello)
```

fitted.values()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** valori fittati
- **Formula:**

$$\hat{y}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> fitted.values(modello)
```

18.3 Adattamento

logLik() s

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** log-verosimiglianza normale
- **Formula:**

$$\hat{\ell}$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> logLik(modello)
```

AIC()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** indice AIC
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> AIC(modello)
```

extractAIC()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** indice AIC generalizzato
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> extractAIC(modello)
```

deviance()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** devianza residua
- **Formula:**

$$D$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> deviance(modello)
```

anova()

- **Package:** `stats`
- **Parametri:**

  `nullo` modello di regressione log-log complementare nullo con n unità

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative con n unità

- **Significato:** anova di regressione
- **Output:**

  `Resid. Df` gradi di libertà

  `Resid. Dev` devianza residua

  `Df` differenza dei gradi di libertà

  `Deviance` differenza tra le devianze residue

  `P(>|Chi|)` p-value

- **Formula:**

  `Resid. Df`

  $$n-1 \quad n-k$$

  `Resid. Dev`

  $$D_{nullo} \quad D$$

  `Df`

  $$1$$

  `Deviance`

  $$c = D_{nullo} - D$$

  `P(>|Chi|)`

  $$P(\chi^2_{k-1} \geq c)$$

- **Esempio:**

18.4 Diagnostica

rstandard()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> rstandard(modello)
```

residuals()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

  `type = deviance / pearson` tipo di residuo

- **Significato:** residui
- **Formula:**

$$\boxed{\texttt{type = deviance}}$$

$$e$$

$$\boxed{\texttt{type = pearson}}$$

$$\frac{y_i - n_i\,\hat{\pi}_i}{\sqrt{n_i\,\hat{\pi}_i\,(1-\hat{\pi}_i)}} \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> residuals(modello,type="deviance")
```

resid()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

  `type = deviance / pearson` tipo di residuo

- **Significato:** residui
- **Formula:**

$$\boxed{\texttt{type = deviance}}$$

$$e$$

$$\boxed{\texttt{type = pearson}}$$

$$\frac{y_i - n_i\,\hat{\pi}_i}{\sqrt{n_i\,\hat{\pi}_i\,(1-\hat{\pi}_i)}} \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> resid(modello,type="deviance")
```

df.residual()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** gradi di libertà della devianza residua
- **Formula:**

$$n-k$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> df.residual(modello)
```

hatvalues()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Significato:** valori di leva
- **Formula:**

$$h$$

- **Esempio:**

```
> modello<-glm(formula=cbind(y,Total-y)~x1+x2+x3,family=binomial(link="cloglog"))
> hatvalues(modello)
```

Capitolo 19

Regressione di Poisson

19.1 Simbologia

$$\log(\mu_i) = \beta_1 + \beta_2 x_{i1} + \beta_3 x_{i2} + \cdots + \beta_k x_{ik-1} \qquad Y_i \sim \text{Poisson}(\mu_i) \quad \forall\, i = 1, 2, \ldots, n$$

- numero di conteggi: $y_i \quad \forall\, i = 1, 2, \ldots, n$
- matrice del modello di dimensione $n \times k$: X
- numero di parametri da stimare e rango della matrice del modello: k
- numero di unità: n
- i-esima riga della matrice del modello : $X_i = (1, x_{i1}, x_{i2}, \ldots, x_{ik-1})$
- matrice diagonale dei pesi IWLS di dimensione $n \times n$: $W = \text{diag}(w_1^{-1}, w_2^{-1}, \ldots, w_n^{-1})$
- matrice di proiezione di dimensione $n \times n$: $H = X\,(X^T\,W^{-1}\,X)^{-1}\,X^T\,W^{-1}$
- devianza residua: $D = 2\sum_{i=1}^{n} y_i \log\left(\frac{y_i}{\hat{\mu}_i} + \frac{1}{2}\,\frac{1-\text{sign}(y_i)}{\hat{\mu}_i}\right) = \sum_{i=1}^{n} e_i^2$
- gradi di libertà della devianza residua: $n - k$
- stime: $\hat{\beta}$
- standard error delle stime: $s_{\hat{\beta}} = \sqrt{\text{diag}((X^T\,W^{-1}\,X)^{-1})}$
- z-values delle stime: $z_{\hat{\beta}} = \hat{\beta}\,/\,s_{\hat{\beta}}$
- residui: $e_i \quad \forall\, i = 1, 2, \ldots, n$
- residui standard: $rstandard_i = \frac{e_i}{\sqrt{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$
- valori fittati: $\hat{\mu}_i = \exp\left(X_i\,\hat{\beta}\right) \quad \forall\, i = 1, 2, \ldots, n$
- valori di leva: $h = \text{diag}(H)$
- devianza residua modello nullo: $D_{nullo} = \sum_{i=1}^{n} y_i \log\left(\frac{y_i}{\hat{\mu}} + \frac{1}{2}\,\frac{1-\text{sign}(y_i)}{\hat{\mu}}\right)$ dove $\hat{\mu} = \bar{y}$
- log-verosimiglianza di *Poisson*: $\hat{\ell} = \sum_{i=1}^{n} [y_i \log(\hat{\mu}_i) - \hat{\mu}_i - \log(y_i!)]$

19.2 Stima

glm()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

  `x = T / F` matrice del modello

- **Significato:** analisi di regressione di *Poisson*

- **Output:**

  `coefficients` stime

  `fitted.values` valori fittati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` della devianza residua

  `aic` indice AIC

  `null.deviance` devianza residua modello nullo

  `weights` pesi IWLS

  `prior.weights` pesi iniziali

  `df.residual` gradi di libertà devianza residua

  `df.null` gradi di libertà devianza residua modello nullo

  `y` numero di conteggi

  `x` matrice del modello

- **Formula:**

  `coefficients`

  $$\hat{\beta}$$

  `fitted.values`

  $$\hat{\mu}_i \quad \forall\, i = 1, 2, \ldots, n$$

  `rank`

  $$k$$

  `linear.predictors`

  $$X\,\hat{\beta}$$

  `deviance`

  $$D$$

  `aic`

  $$-2\,\hat{\ell} + 2\,k$$

  `null.deviance`

  $$D_{nullo}$$

  `weights`

  $$w_i \quad \forall\, i = 1, 2, \ldots, n$$

  `prior.weights`

  $$\underbrace{1, 1, \ldots, 1}_{n \text{ volte}}$$

  `df.residual`

  $$n - k$$

  `df.null`

  $$n - 1$$

  `y`

  $$y_i \quad \forall\, i = 1, 2, \ldots, n$$

  `x`

  $$X$$

- **Esempio:**

```
> glm(formula=y~x1+x2+x3,family=poisson(link="log"),x=T)
```

summary.glm()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** analisi di regressione di *Poisson*
- **Output:**

  `deviance` devianza residua

  `aic` indice AIC

  `df.residual` gradi di libertà devianza residua

  `null.deviance` devianza residua modello nullo

  `df.null` gradi di libertà devianza residua modello nullo

  `deviance.resid` residui di devianza

  `coefficients` stima puntuale, standard error, z-value, p-value

  `cov.unscaled` matrice di covarianza delle stime non scalata

  `cov.scaled` matrice di covarianza delle stime scalata

- **Formula:**

  `deviance`

  $$D$$

  `aic`

  $$-2\,\hat{\ell} + 2\,k$$

  `df.residual`

  $$n - k$$

  `df.null`

  $$n - 1$$

  `deviance.resid`

  $$e$$

  `coefficients`

  $$\hat{\beta}_j \quad s_{\hat{\beta}_j} \quad z_{\hat{\beta}_j} \quad p\text{-value} = 2\,\Phi(-\,|\,z_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1, 2, \ldots, k$$

  `cov.unscaled`

  $$(X^T\,W^{-1}\,X)^{-1}$$

  `cov.scaled`

  $$(X^T\,W^{-1}\,X)^{-1}$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> summary.glm(modello)
```

glm.fit()

- **Package:** `stats`
- **Parametri:**

  `x` matrice del modello
  `y` numero di conteggi

- **Significato:** analisi di regressione di *Poisson*
- **Output:**

  `coefficients` stime
  `fitted.values` valori fittati
  `rank` rango della matrice del modello
  `linear.predictors` predittori lineari
  `deviance` della devianza residua
  `aic` indice *AIC*
  `null.deviance` devianza residua modello nullo
  `weights` pesi IWLS
  `prior.weights` pesi iniziali
  `df.residual` gradi di libertà devianza residua
  `df.null` gradi di libertà devianza residua modello nullo
  `y` numero di conteggi

- **Formula:**

  `coefficients`
  $$\hat{\beta}$$

  `fitted.values`
  $$\hat{\mu}_i \quad \forall\, i = 1, 2, \ldots, n$$

  `rank`
  $$k$$

  `linear.predictors`
  $$X\,\hat{\beta}$$

  `deviance`
  $$D$$

  `aic`
  $$-2\,\hat{\ell} + 2\,k$$

  `null.deviance`
  $$D_{nullo}$$

  `weights`
  $$w_i \quad \forall\, i = 1, 2, \ldots, n$$

  `prior.weights`
  $$\underbrace{1, 1, \ldots, 1}_{n \text{ volte}}$$

  `df.residual`
  $$n - k$$

  `df.null`
  $$n - 1$$

  `y`
  $$y_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Esempio:**

```
> glm.fit(formula=y~x1+x2+x3)
```

vcov()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** matrice di covarianza delle stime
- **Formula:**

$$s^2 \, (X^T \, W^{-1} \, X)^{-1}$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> vcov(modello)
```

coef()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** stime
- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> coef(modello)
```

coefficients()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** stime
- **Formula:**

$$\hat{\beta}$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> coefficients(modello)
```

fitted()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** valori fittati
- **Formula:**

$$\hat{\mu}$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> fitted(modello)
```

fitted.values()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** valori fittati
- **Formula:**

$$\hat{\mu}$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> fitted.values(modello)
```

19.3 Adattamento

logLik()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** log-verosimiglianza normale
- **Formula:**

$$\hat{\ell}$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> logLik(modello)
```

AIC()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> AIC(modello)
```

extractAIC()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** indice *AIC* generalizzato
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> extractAIC(modello)
```

deviance()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** devianza residua
- **Formula:**

$$D$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> deviance(modello)
```

anova()

- **Package:** `stats`
- **Parametri:**

  `mod` modello di regressione di *Poisson* nullo con n unità

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative con n unità

- **Significato:** anova di regressione
- **Output:**

  `Resid. Df` gradi di libertà

  `Resid. Dev` devianza residua

  `Df` differenza dei gradi di libertà

  `Deviance` differenza tra le devianze residue

  `P(>|Chi|)` p-value

- **Formula:**

  `Resid. Df`

  $$n-1 \quad n-k$$

  `Resid. Dev`

  $$D_{nullo} \quad D$$

  `Df`

  $$1$$

  `Deviance`

  $$c = D_{nullo} - D$$

  `P(>|Chi|)`

  $$P(\chi^2_{k-1} \geq c)$$

- **Esempio:**

19.4 Diagnostica

rstandard()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> rstandard(modello)
```

residuals()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

  `type = deviance / pearson` tipo di residuo

- **Significato:** residui
- **Formula:**

$$\boxed{\texttt{type = deviance}}$$

$$e$$

$$\boxed{\texttt{type = pearson}}$$

$$\frac{y_i - \hat{\mu}_i}{\sqrt{\hat{\mu}_i}} \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> residuals(modello,type="deviance")
```

resid()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

  `type = deviance / pearson` tipo di residuo

- **Significato:** residui
- **Formula:**

$$\boxed{\texttt{type = deviance}}$$

$$e$$

$$\boxed{\texttt{type = pearson}}$$

$$\frac{y_i - \hat{\mu}_i}{\sqrt{\hat{\mu}_i}} \quad \forall i = 1, 2, \ldots, n$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> resid(modello,type="deviance")
```

df.residual()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** gradi di libertà della devianza residua
- **Formula:**

$$n-k$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> df.residual(modello)
```

hatvalues()

- **Package:** `stats`
- **Parametri:**

  `formula` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Significato:** valori di leva
- **Formula:**

$$h$$

- **Esempio:**

```
> modello<-glm(formula=y~x1+x2+x3,family=poisson(link="log"))
> hatvalues(modello)
```

Parte VI

Appendice

Appendice A

Packages

- ○ `base` The R Base Package
- • `boot` Bootstrap R (S-Plus) Functions (Canty)
- • `car` Companion to Applied Regression
- • `BSDA` Basic Statistics and Data Analysis
- ○ `datasets` The R Datasets Package
- • `distributions` Probability distributions based on TI-83 Plus
- • `e1071` Misc Functions of the Department of Statistics (e1071), TU Wien
- • `faraway` Functions and datasets for books by Julian Faraway
- • `fBasics` Financial Software Collection
- • `foreign` Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...
- ○ `graphics` The R Graphics Package
- ○ `grDevices` The R Graphics Devices and Support for Colours and Fonts
- • `gtools` Various R programming tools
- • `labstatR` Libreria del Laboratorio di Statistica con R
- • `leaps` regression subset selection
- • `lmtest` Testing Linear Regression Models
- • `MASS` Main Package of Venables and Ripley's MASS
- • `Matrix` A Matrix package for R

- `MCMCpack` Markov chain Monte Carlo (MCMC) Package
- `methods` Formal Methods and Classes
- `moments` Moments, skewness, kurtosis and related tests
- `mvtnorm` Multivariate Normal and T Distribution
- `nlme` Linear and nonlinear mixed effects models
- `nortest` Tests for Normality
- `Rcmdr` R Commander
- `sigma2tools` Test of hypothesis about sigma2
- `stats` The R Stats Package
- `strucchange` Testing for Structural Change
- `SuppDists` Supplementary distributions
- `tseries` Time series analysis and computational finance
- `utils` The R Utils Package

Legenda

- *Package* automaticamente installato ed avviato
- *Package* da installare

Download Packages

Bibliografia

[1] C. Gaetan e N. Sartori A. Brazzale, M. Chiogna. Laboratorio di R, Materiale didattico per i laboratori del corso di Modelli Statistici I. Published on the URL: http://www.isib.cnr.it/~brazzale/ModStatI/, 2001.

[2] C. Agostinelli. Introduzione ad R. Published on the URL: http://www.dst.unive.it/~laboratorior/doc/materiale/unaintroduzioneadR.pdf, 2000.

[3] Saghir Bashir. Getting Started in R. Published on the URL: http://www.sbtc.ltd.uk/notes/Rintro.pdf, 2004.

[4] R. Boggiani. Introduzione ad R. Published on the URL: http://digilander.libero.it/robicox/manuali/pdf/mainr.pdf, 2004.

[5] F. Crivellari. *Analisi Statistica dei dati con R.* APOGEO Edizioni, Milano, Italia, 2006.

[6] G. D'Agostini. Il linguaggio R: Un invito ad approfondire. Published on the URL: http://www.roma1.infn.it/~dagos/R/R.pdf, Università degli Studi di Roma La Sapienza e INFN, 2005.

[7] P. Dalgaard. *Introductory Statistics with R.* Springer-Verlag, New York, 2002.

[8] J. Faraway. Practical Regression and Anova using R. Published on the URL: http://www.stat.lsa.umich.edu/~faraway/book/pra.pdf, 2002.

[9] J. Fox. *An R and S-Plus Companion to Applied Regression.* SAGE Pubblications, Thousand Oaks, California, 2002.

[10] Christpher Green. The Stat 390 R Primer. Published on the URL: http://www.stat.washington.edu/cggreen/rprimer/rprimer.pdf.

[11] Søren Højsgaard. R - In Two HouRs – a very brief introduction. Published on the URL: http://gbi.agrsci.dk/statistics/courses/phd05/material/src/R-2hours-Notes.pdf, Biometry Research Unit, Danish Institute of Agricultural Sciences, 2005.

[12] Dong-Yun Kim. R Tutorial. Published on the URL: http://www.math.ilstu.edu/dhkim/Rstuff/Rtutor.html, Department of Mathematics Illinois State University, 2004.

[13] J. Lemon. Kickstarting R. Published on the URL: http://www.cran.r-project.org/doc/contrib/Lemon-kickstart/index.html, 2005.

[14] J. H. Maindonald. Using R for Data Analysis and Graphics Introduction, Code and Commentary. Published on the URL: http://www.cran.r-project.org/doc/contrib/usingR.pdf, 2004.

[15] Angelo M. Mineo. Una guida all'utilizzo dell'ambiente statistico R. Published on the URL: http://www.cran.r-project.org/doc/contrib/Mineo-dispensaR.pdf, 2003.

[16] Vito M. R. Muggeo. Il linguaggio R: concetti introduttivi ed esempi. Published on the URL: http://www.cran.r-project.org/doc/contrib/nozioniR.pdf, 2002.

[17] W. J. Owen. The R Guide. Published on the URL: http://www.mathcs.richmond.edu/~wowen/TheRGuide.pdf, 2006.

[18] Emmanuel Paradis. R for beginners. Published on the URL: http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf, 2002.

[19] F. Parpinel. La statistica applicata attraverso l'uso del programma R. Published on the URL: http://venus.unive.it/statcomp/r/man_Parpinel.pdf, 2000.

[20] S. Polettini. Introduzione ad R. Published on the URL: http://www.dipstat.unina.it/stat_appl/labo1.pdf, 2004.

[21] A. Pollice. La statistica applicata attraverso l'uso del programma R. Published on the URL: http://www.dip-statistica.uniba.it/html/docenti/pollice/materiale.htm, Dipartimento di Scienze Statistiche, Università di Bari, 2000.

[22] V. Ricci. ANALISI DELLE SERIE STORICHE CON R. Published on the URL: http://www.cran.r-project.org/doc/contrib/Ricci-ts-italian.pdf, 2004.

[23] Andrew Robinson. Objects in R. Published on the URL: http://www.forestry.ubc.ca/biometrics/documents/R-Workshop/objects.pdf, 2006.

[24] Theresa Scott. An Introduction to R. Published on the URL: http://www.mc.vanderbilt.edu/gcrc/workshop_files/2004-08-20.pdf, 2004.

[25] L. Scrucca. Note sul linguaggio e ambiente statistico R. Published on the URL: http://www.stat.unipg.it/~luca/LabStat/R-note.pdf, Dipartimento di Scienze Statistiche, Università degli Studi di Perugia, 2005.

[26] L. Soliani. Manuale di Statistica per la Ricerca e la Professione. Published on the URL: http://www.dsa.unipr.it/soliani/soliani.html, 2005.

[27] A. Tancredi. Inferenza statistica in applicazioni economiche ed aziendali. Published on the URL: http://geostasto.eco.uniroma1.it/utenti/tancredi/isaea1-2x1.pdf, Università degli Studi di Roma La Sapienza, 2005.

[28] W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, New York, 2002.

[29] John Verzani. Using R for Introductory Statistics. Published on the URL: http://www.cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf, 2002.

Indice analitico