Formulario di Statistica con

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

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

Fabio Frascati[1]

R version 2.7.0 (2008-04-22)

Work in progress!

6 settembre 2008

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

Indice

Parte I

Matematica ed algebra lineare

Capitolo 1

Background

1.1 Operatori matematici

+

- **Package:** base
- **Description:** addizione
- **Example:**

```
> 1 + 2

[1] 3

> x <- c(1, 2, 3, 4, 5)
> y <- c(1.2, 3.4, 5.2, 3.5, 7.8)
> x + y

[1]  2.2  5.4  8.2  7.5 12.8

> x <- c(1, 2, 3, 4, 5)
> x + 10

[1] 11 12 13 14 15
```

−

- **Package:** base
- **Description:** sottrazione
- **Example:**

```
> 1.2 - 6.7

[1] -5.5

> x <- c(1, 2, 3, 4, 5)
> y <- c(1.2, 3.4, 5.2, 3.5, 7.8)
> x - y

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

> x <- c(1, 2, 3, 4, 5)
> x - 10

[1] -9 -8 -7 -6 -5
```

```
> Inf - Inf

[1] NaN

> --3

[1] 3
```

*

- **Package:** base
- **Description:** moltiplicazione
- **Example:**

```
> 2.3 * 4

[1] 9.2

> x <- c(1.2, 3.4, 5.6, 7.8, 0, 9.8)
> 3 * x

[1]  3.6 10.2 16.8 23.4  0.0 29.4

> x <- c(1, 2, 3, 4, 5, 6, 7)
> y <- c(-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
- **Description:** rapporto
- **Example:**

```
> 21/7

[1] 3

> x <- c(1.2, 3.4, 5.6, 7.8, 0, 9.8)
> x/2

[1] 0.6 1.7 2.8 3.9 0.0 4.9

> 2/0

[1] Inf

> -1/0

[1] -Inf

> 0/0
```

```
[1] NaN
```

```
> Inf/Inf
```

```
[1] NaN
```

```
> Inf/0
```

```
[1] Inf
```

```
> -Inf/0
```

```
[1] -Inf
```

```
> x <- c(1, 2, 3, 4, 5, 6, 7)
> y <- c(-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
- **Description:** elevamento a potenza
- **Example:**

```
> 2**4
```

```
[1] 16
```

```
> x <- c(1.2, 3.4, 5.6, 7.8, 0.0, 9.8)
> x**2
```

```
[1]  1.44 11.56 31.36 60.84  0.00 96.04
```

```
> x <- c(1, 2, 3, 4)
> y <- c(-3.2, -2.2, -1.2, -0.2)
> y**x
```

```
[1] -3.2000  4.8400 -1.7280  0.0016
```

^

- **Package:** base
- **Description:** elevamento a potenza
- **Example:**

```
> 2^4

[1] 16

> x <- c(1.2, 3.4, 5.6, 7.8, 0, 9.8)
> x^2

[1]  1.44 11.56 31.36 60.84  0.00 96.04

> x <- c(1, 2, 3, 4)
> y <- c(-3.2, -2.2, -1.2, -0.2)
> y^x

[1] -3.2000  4.8400 -1.7280  0.0016
```

%/%

- **Package:** base
- **Description:** quoziente intero della divisione
- **Example:**

```
> 22.6%/%3.4

[1] 6

> 23%/%3

[1] 7
```

%%

- **Package:** base
- **Description:** resto della divisione (modulo)
- **Example:**

```
> 22.6%%3.4

[1] 2.2

> 23%%3

[1] 2
```

1.2 Operatori relazionali

<

- **Package:** base
- **Description:** minore
- **Example:**

```
> 1 < 2

[1] TRUE

> x <- c(0.11, 1.2, 2.3, 4.5)
> x < 2.4

[1]  TRUE  TRUE  TRUE FALSE
```

>

- **Package:** base
- **Description:** maggiore
- **Example:**

```
> 3 > 1.2

[1] TRUE

> x <- c(0.11, 1.2, 2.3, 4.5)
> x > 2.4

[1] FALSE FALSE FALSE  TRUE
```

<=

- **Package:** base
- **Description:** minore od uguale
- **Example:**

```
> 3.4 <= 8.5

[1] TRUE

> x <- c(0.11, 1.2, 2.3, 4.5)
> x <= 2.4

[1]  TRUE  TRUE  TRUE FALSE
```

>=

- **Package:** base
- **Description:** maggiore od uguale
- **Example:**

```
> 3.4 >= 5.4

[1] FALSE

> x <- c(0.11, 1.2, 2.3, 5.4)
> x >= 5.4

[1] FALSE FALSE FALSE  TRUE
```

!=

- **Package:** base
- **Description:** diverso
- **Example:**

```
> 2 != 3

[1] TRUE

> x <- c(0.11, 1.2, 2.3, 5.4)
> x != 5.4

[1]  TRUE  TRUE  TRUE FALSE
```

==

- **Package:** base
- **Description:** uguale
- **Example:**

```
> 4 == 4

[1] TRUE

> x <- c(0.11, 1.2, 2.3, 5.4)
> x == 5.4

[1] FALSE FALSE FALSE  TRUE

> TRUE == 1

[1] TRUE

> FALSE == 0

[1] TRUE
```

1.3 Operatori logici

&

- **Package:** base
- **Description:** AND termine a termine
- **Example:**

```
> 1 & 5

[1] TRUE

> x <- c(0.11, 1.2, 2.3, 4.5, 0)
> x & 3

[1]  TRUE  TRUE  TRUE  TRUE FALSE
```

&&

- **Package:** base
- **Description:** AND si arresta al primo elemento che soddisfa la condizione
- **Example:**

```
> 1 && 5

[1] TRUE

> x <- c(0.11, 1.2, 2.3, 4.5, 0)
> x && 3

[1] TRUE

> x <- c(0, 1.2, 2.3, 4.5, 0)
> x && 3

[1] FALSE
```

|

- **Package:** base
- **Description:** OR termine a termine
- **Example:**

```
> 5 | 0

[1] TRUE

> x <- c(0.11, 1.2, 2.3, 4.5, 0)
> x | 0

[1]  TRUE  TRUE  TRUE  TRUE FALSE
```

||

- **Package:** base
- **Description:** OR si arresta al primo elemento che soddisfa la condizione
- **Example:**

```
> 5 || 0

[1] TRUE

> x <- c(0.11, 1.2, 2.3, 4.5, 0)
> x || 3

[1] TRUE

> x <- c(0, 1.2, 2.3, 4.5, 0)
> x || 0

[1] FALSE
```

xor()

- **Package:** base
- **Description:** EXCLUSIVE OR termine a termine
- **Example:**

```
> xor(4, 5)

[1] FALSE

> x <- c(0.11, 1.2, 2.3, 4.5, 0)
> xor(x, 3)

[1] FALSE FALSE FALSE FALSE  TRUE
```

!

- **Package:** base
- **Description:** NOT
- **Example:**

```
> !8

[1] FALSE

> x <- c(0.11, 1.2, 2.3, 4.5, 0)
> !x

[1] FALSE FALSE FALSE FALSE  TRUE
```

1.4 Funzioni di base

sum()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

- **Description:** somma
- **Formula:**

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

- **Example:**

```
> x <- c(1.2, 2, 3)
> 1.2 + 2 + 3

[1] 6.2

> sum(x)

[1] 6.2

> x <- c(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
- **Input:**

  x vettore numerico di dimensione n

- **Description:** prodotto
- **Formula:**

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

- **Example:**

```
> x <- c(1, 2, 3.2)
> 1 * 2 * 3.2

[1] 6.4

> prod(x)

[1] 6.4

> x <- c(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
- **Input:**

  x valore numerico

- **Description:** valore assoluto
- **Formula:**

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

- **Example:**

```
> abs(x = 1.3)
```

```
[1] 1.3
```

```
> abs(x = 0)
```

```
[1] 0
```

```
> abs(x = -2.3)
```

```
[1] 2.3
```

```
> abs(x = 3 + 4i)
```

```
[1] 5
```

```
> Mod(x = 3 + 4i)
```

```
[1] 5
```

- **Note:** Equivale alla funzione Mod().

sign()

- **Package:** base
- **Input:**

  x valore numerico

- **Description:** segno
- **Formula:**

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

- **Example:**

```
> sign(x = 1.2)

[1] 1

> sign(x = 0)

[1] 0

> sign(x = -1.2)

[1] -1
```

sqrt()

- **Package:** base
- **Input:**

  x valore numerico tale che $x > 0$

- **Description:** radice quadrata
- **Formula:**

$$\sqrt{x}$$

- **Example:**

```
> sqrt(x = 2)

[1] 1.414214

> sqrt(x = 3.5)

[1] 1.870829

> sqrt(x = -9)

[1] NaN

> sqrt(x = -9 + 0i)

[1] 0+3i
```

1.5 Funzioni insiemistiche

union()

- **Package:** base
- **Input:**

  x vettore alfanumerico di dimensione n

  y vettore alfanumerico di dimensione m

- **Description:** unione
- **Formula:**

$$x \cup y$$

- **Example:**

```
> x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
> y <- c(1, 2, 6, 11)
> union(x, y)

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

> x <- c("a", "b", "c", "d", "e", "f", "g")
> y <- c("a", "e", "f", "h")
> union(x, y)

[1] "a" "b" "c" "d" "e" "f" "g" "h"
```

intersect()

- **Package:** base
- **Input:**

  x vettore alfanumerico di dimensione n

  y vettore alfanumerico di dimensione m

- **Description:** intersezione
- **Formula:**

$$x \cap y$$

- **Example:**

```
> x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
> y <- c(1, 2, 6, 11)
> intersect(x, y)

[1] 1 2 6

> x <- c("a", "b", "c", "d", "e", "f", "g")
> y <- c("a", "e", "f", "h")
> intersect(x, y)

[1] "a" "e" "f"
```

setdiff()

- **Package:** base
- **Input:**

  `x` vettore alfanumerico di dimensione n
  `y` vettore alfanumerico di dimensione m

- **Description:** differenza
- **Formula:**

$$x \setminus y$$

- **Example:**

```
> x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
> y <- c(1, 2, 6, 11)
> setdiff(x, y)

[1]  3  4  5  7  8  9 10

> x <- c("a", "b", "c", "d", "e", "f", "g")
> y <- c("a", "e", "f", "h")
> setdiff(x, y)

[1] "b" "c" "d" "g"
```

is.element()

- **Package:** base
- **Input:**

  `el` valore x alfanumerico
  `set` vettore y alfanumerico di dimensione n

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

$$x \in y$$

- **Example:**

```
> x <- 2
> y <- c(1, 2, 6, 11)
> is.element(el = x, set = y)

[1] TRUE

> x <- 3
> y <- c(1, 2, 6, 11)
> is.element(el = x, set = y)

[1] FALSE

> x <- "d"
> y <- c("a", "b", "c", "d", "e", "f", "g")
> is.element(el = x, set = y)

[1] TRUE

> x <- "h"
> y <- c("a", "b", "c", "d", "e", "f", "g")
> is.element(el = x, set = y)

[1] FALSE
```

%in%

- **Package:** base
- **Input:**

  x valore alfanumerico

  y vettore alfanumerico di dimensione n
- **Description:** appartenenza di x all'insieme y
- **Formula:**

$$x \in y$$

- **Example:**

```
> x <- 2
> y <- c(1, 2, 6, 11)
> x %in% y

[1] TRUE

> x <- 3
> y <- c(1, 2, 6, 11)
> x %in% y

[1] FALSE

> x <- "d"
> y <- c("a", "b", "c", "d", "e", "f", "g")
> x %in% y

[1] TRUE

> x <- "h"
> y <- c("a", "b", "c", "d", "e", "f", "g")
> x %in% y

[1] FALSE
```

setequal()

- **Package:** base
- **Input:**

  x vettore alfanumerico di dimensione n

  y vettore alfanumerico di dimensione m
- **Description:** uguaglianza
- **Formula:**

$$x = y \Leftrightarrow \left\{ \begin{array}{l} x \subseteq y \\ y \subseteq x \end{array} \right.$$

- **Example:**

```
> x <- c(1, 4, 5, 6, 8, 77)
> y <- c(1, 1, 1, 4, 5, 6, 8, 77)
> setequal(x, y)

[1] TRUE
```

```
> x <- c("a", "b")
> y <- c("a", "b", "a", "b", "a", "b", "a")
> setequal(x, y)

[1] TRUE
```

1.6 Funzioni indice

which()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n
- **Description:** indici degli elementi di x che soddisfano ad una condizione fissata
- **Example:**

```
> x <- c(1.2, 4.5, -1.3, 4.5)
> which(x > 2)

[1] 2 4

> x <- c(1.2, 4.5, -1.3, 4.5)
> which((x >= -1) & (x < 5))

[1] 1 2 4

> x <- c(1.2, 4.5, -1.3, 4.5)
> which((x >= 3.6) | (x < -1.6))

[1] 2 4

> x <- c(1.2, 4.5, -1.3, 4.5)
> x[x < 4]

[1]  1.2 -1.3

> x[which(x < 4)]

[1]  1.2 -1.3
```

which.min()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n
- **Description:** indice del primo elemento minimo di x
- **Example:**

```
> x <- c(1.2, 1, 2.3, 4, 1, 4)
> min(x)

[1] 1
```

```
> which(x == min(x))[1]

[1] 2

> which.min(x)

[1] 2

> x <- c(1.2, 4.5, -1.3, 4.5)
> min(x)

[1] -1.3

> which(x == min(x))[1]

[1] 3

> which.min(x)

[1] 3
```

which.max()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

- **Description:** indice del primo elemento massimo di x
- **Example:**

```
> x <- c(1.2, 1, 2.3, 4, 1, 4)
> max(x)

[1] 4

> which(x == max(x))[1]

[1] 4

> which.max(x)

[1] 4

> x <- c(1.2, 4.5, -1.3, 4.5)
> max(x)

[1] 4.5

> which(x == max(x))[1]

[1] 2

> which.max(x)

[1] 2
```

1.7 Funzioni combinatorie

choose()

- **Package:** base
- **Input:**

  n valore naturale

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

- **Description:** coefficiente binomiale
- **Formula:**

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

- **Example:**

```
> 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
- **Input:**

  n valore naturale

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

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

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

- **Example:**

```
> 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
- **Input:**

  x valore naturale

- **Description:** fattoriale
- **Formula:**

$$x\,!$$

- **Example:**

```
> 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
- **Input:**

  x valore naturale

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

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

- **Example:**

```
> 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 dirette

sin()

- **Package:** base
- **Input:**

  x valore numerico

- **Description:** seno
- **Formula:**

$$\sin(x)$$

- **Example:**

```
> sin(x = 1.2)

[1] 0.932039

> sin(x = pi)

[1] 1.224606e-16
```

cos()

- **Package:** base
- **Input:**

  x valore numerico

- **Description:** coseno
- **Formula:**

$$\cos(x)$$

- **Example:**

```
> cos(x = 1.2)

[1] 0.3623578

> cos(x = pi/2)

[1] 6.123032e-17
```

tan()

- **Package:** base
- **Input:**

  x valore numerico

- **Description:** tangente
- **Formula:**

$$\tan(x) = \frac{\sin(x)}{\cos(x)}$$

- **Example:**

```
> tan(x = 1.2)

[1] 2.572152

> tan(x = pi)

[1] -1.224606e-16

> tan(x = 2.3)

[1] -1.119214

> sin(x = 2.3)/cos(x = 2.3)

[1] -1.119214
```

1.9 Funzioni trigonometriche inverse

asin()

- **Package:** base
- **Input:**

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

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

$$\arcsin(x)$$

- **Example:**

```
> asin(x = 0.9)

[1] 1.119770

> asin(x = -1)

[1] -1.570796
```

acos()

- **Package:** base
- **Input:**

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

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

$$\arccos(x)$$

- **Example:**

```
> acos(x = 0.9)

[1] 0.4510268

> acos(x = -1)

[1] 3.141593
```

atan()

- **Package:** base
- **Input:**

  x valore numerico

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

$$\arctan(x)$$

- **Example:**

```
> atan(x = 0.9)
```

```
[1] 0.7328151
```

```
> atan(x = -34)
```

```
[1] -1.541393
```

atan2()

- **Package:** base
- **Input:**

  y valore numerico di ordinata
  x valore numerico di ascissa

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

$$\arctan(x)$$

- **Example:**

```
> atan2(y = -2, x = 0.9)
```

```
[1] -1.147942
```

```
> atan2(y = -1, x = -1)
```

```
[1] -2.356194
```

1.10 Funzioni iperboliche dirette

sinh()

- **Package:** base
- **Input:**

  x valore numerico

- **Description:** seno iperbolico
- **Formula:**

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

- **Example:**

```
> 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
- **Input:**

  x valore numerico

- **Description:** coseno iperbolico
- **Formula:**

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

- **Example:**

```
> 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
- **Input:**

  x valore numerico

- **Description:** tangente iperbolica
- **Formula:**

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

- **Example:**

```
> 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

> tanh(x = 2.3)

[1] 0.9800964

> sinh(x = 2.3)/cosh(x = 2.3)

[1] 0.9800964
```

1.11 Funzioni iperboliche inverse

asinh()

- **Package:** base
- **Input:**

  x valore numerico

- **Description:** inversa seno iperbolico
- **Formula:**

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

- **Example:**

```
> asinh(x = 2.45)

[1] 1.628500

> asinh(x = 3.7)

[1] 2.019261
```

acosh()

- **Package:** base
- **Input:**

  x valore numerico tale che $x \geq 1$

- **Description:** inversa coseno iperbolico
- **Formula:**

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

- **Example:**

```
> acosh(x = 2.45)

[1] 1.544713

> acosh(x = 3.7)

[1] 1.982697
```

atanh()

- **Package:** base
- **Input:**

  x valore numerico tale che $|x| < 1$
- **Description:** inversa tangente iperbolica
- **Formula:**

$$\text{arctanh}(x) = \frac{1}{2} \log \left(\frac{1 + x}{1 - x} \right)$$

- **Example:**

```
> x <- 0.45
> 0.5 * log((1 + x)/(1 - x))

[1] 0.4847003

> atanh(x = 0.45)

[1] 0.4847003

> x <- 0.7
> 0.5 * log((1 + x)/(1 - x))

[1] 0.8673005

> atanh(x = 0.7)

[1] 0.8673005
```

1.12 Funzioni esponenziali e logaritmiche

exp()

- **Package:** base
- **Input:**

  x valore numerico
- **Description:** esponenziale
- **Formula:**

$$e^x$$

- **Example:**

```
> exp(x = 1.2)

[1] 3.320117

> exp(x = 0)

[1] 1
```

expm1()

- **Package:** base
- **Input:**

  x valore numerico
- **Description:** esponenziale
- **Formula:**

$$e^x - 1$$

- **Example:**

```
> 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
- **Input:**

  x valore numerico tale che $x > 0$
- **Description:** logaritmo di x in base 2
- **Formula:**

$$\log_2(x)$$

- **Example:**

```
> log2(x = 1.2)

[1] 0.2630344

> log2(x = 8)

[1] 3

> log2(x = -1.2)

[1] NaN
```

log10()

- **Package:** base
- **Input:**

  x valore numerico tale che $x > 0$

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

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

- **Example:**

```
> log10(x = 1.2)

[1] 0.07918125

> log10(x = 1000)

[1] 3

> log10(x = -6.4)

[1] NaN
```

log()

- **Package:** base
- **Input:**

  x valore numerico tale che $x > 0$

  base il valore b tale che $b > 0$

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

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

- **Example:**

```
> 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
```

logb()

- **Package:** base
- **Input:**

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

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

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

$$\log_b(x)$$

- **Example:**

```
> logb(x = 2, base = 4)

[1] 0.5

> logb(x = 8, base = 2)

[1] 3

> logb(x = -1.2, base = 2)

[1] NaN
```

log1p()

- **Package:** base
- **Input:**

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

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

$$\log(x + 1)$$

- **Example:**

```
> 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

> log1p(x = -1)

[1] -Inf

> log1p(x = -1.2)

[1] NaN
```

1.13 Funzioni di successione

:

- **Package:** base
- **Description:** successione con intervallo unitario
- **Example:**

```
> 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
- **Input:**

  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

- **Description:** replicazioni
- **Example:**

```
> 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(8.1, 6.7, 10.2), times = c(1, 2, 3))

[1]  8.1  6.7  6.7 10.2 10.2 10.2

> 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 = TRUE, times = 5)

[1] TRUE TRUE TRUE TRUE TRUE

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

 [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
```

- **Note:** Il parametro `each` ha precedenza sul parametro `times`.

rep.int()

- **Package:** `base`
- **Input:**

  `x` vettore alfanumerico di dimensione n

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

- **Description:** replicazioni
- **Example:**

```
> 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.int(x = TRUE, times = 5)

[1] TRUE TRUE TRUE TRUE TRUE
```

sequence()

- **Package:** base
- **Input:**

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

- **Description:** serie di sequenze di interi dove ciascuna sequenza termina con i numeri naturali passati come argomento
- **Example:**

```
> 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
- **Input:**

  `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 $1, 2, \ldots, n$

- **Description:** successione
- **Example:**

```
> 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 <- c(1.5, 6.4, 9.6, 8.8)
> n <- 4
> 1:n

[1] 1 2 3 4

> seq(along.with = x)

[1] 1 2 3 4

> x <- c(1.5, 6.4, 9.6, 8.8)
> seq(from = 88, to = 50, along.with = x)

[1] 88.00000 75.33333 62.66667 50.00000

> seq(from = 88, to = 50, length.out = length(x))

[1] 88.00000 75.33333 62.66667 50.00000

> 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
```

seq_along()

- **Package:** base
- **Input:**

  along.with vettore numerico x di dimensione n

- **Description:** sequenza di valori naturali $1, 2, \ldots, n$
- **Example:**

```
> x <- c(1.2, 2.3, 3.4, 4.5, 5.6, 6.7)
> n <- 6
> seq_along(along.with = x)

[1] 1 2 3 4 5 6

> x <- c(1.5, 6.4, 9.6, 8.8)
> n <- 4
> seq_along(along.with = x)

[1] 1 2 3 4
```

seq_len()

- **Package:** base
- **Input:**

  `length.out` valore n naturale

- **Description:** sequenza di valori naturali $1, 2, \ldots, n$
- **Example:**

```
> n <- 6
> seq_len(length.out = 6)

[1] 1 2 3 4 5 6

> n <- 4
> seq_len(length.out = 4)

[1] 1 2 3 4
```

1.14 Funzioni di ordinamento

sort()

- **Package:** base
- **Input:**

  `x` vettore numerico di dimensione n

  `decreasing = TRUE / FALSE` decremento oppure incremento

  `index.return = TRUE / FALSE` vettore indici ordinati

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

  `x` vettore ordinato

  `ix` vettore indici ordinati

- **Formula:**

  `x`

  `decreasing = TRUE`

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

  `decreasing = FALSE`

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

- **Example:**

```
> x <- c(1.2, 2.3, 4.21, 0, 2.1, 3.4)
> sort(x, decreasing = TRUE, index.return = FALSE)

[1] 4.21 3.40 2.30 2.10 1.20 0.00

> x <- c(1.2, 2.3, 4.21, 0, 2.1, 3.4)
> res <- sort(x, decreasing = TRUE, index.return = TRUE)
> 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[res$ix]

[1] 4.21 3.40 2.30 2.10 1.20 0.00

> x <- c(1.2, 2.3, 4.21, 0, 2.1, 3.4)
> sort(x, decreasing = FALSE, index.return = FALSE)

[1] 0.00 1.20 2.10 2.30 3.40 4.21

> x <- c(1.2, 2.3, 4.21, 0, 2.1, 3.4)
> res <- sort(x, decreasing = FALSE, index.return = TRUE)
> res$x

[1] 0.00 1.20 2.10 2.30 3.40 4.21

> res$ix

[1] 4 1 5 2 6 3

> x[res$ix]

[1] 0.00 1.20 2.10 2.30 3.40 4.21

> x <- c(1.2, 4.2, 4.5, -5.6, 6.5, 1.2)
> sort(x, decreasing = TRUE)

[1]  6.5  4.5  4.2  1.2  1.2 -5.6

> rev(sort(x))

[1]  6.5  4.5  4.2  1.2  1.2 -5.6
```

- **Note:** Equivale alla funzione `order()` quando `index.return = TRUE`.

rev()

- **Package:** `base`
- **Input:**

  `x` vettore numerico di dimensione n
- **Description:** elementi di un vettore in ordine invertito
- **Formula:**

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

- **Example:**

```
> x <- c(1.2, 2.3, 4.21, 0, 2.1, 3.4)
> rev(x)

[1] 3.40 2.10 0.00 4.21 2.30 1.20

> x <- c(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
- **Input:**

  `x` vettore numerico di dimensione n

  `decreasing = TRUE / FALSE` decremento oppure incremento

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

```
> x <- c(1.2, 2.3, 4.21, 0, 2.1, 3.4)
> order(x, decreasing = FALSE)

[1] 4 1 5 2 6 3

> x <- c(1.2, 2.3, 4.21, 0, 2.1, 3.4)
> order(x, decreasing = TRUE)

[1] 3 6 2 5 1 4

> x <- c(1.6, 6.8, 7.7, 7.2, 5.4, 7.9, 8, 8, 3.4, 12)
> sort(x, decreasing = FALSE)

 [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 = FALSE)]

 [1]  1.6  3.4  5.4  6.8  7.2  7.7  7.9  8.0  8.0 12.0
```

rank()

- **Package:** base
- **Input:**

  `x` vettore numerico di dimensione n

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

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

```
> x <- c(1.2, 2.3, 4.5, 2.3, 4.5, 6.6, 1.2, 3.4)
> rank(x, ties.method = "average")

[1] 1.5 3.5 6.5 3.5 6.5 8.0 1.5 5.0

> x <- c(1.2, 2.3, 4.21, 0, 2.1, 3.4)
> rank(x, ties.method = "average")

[1] 2 4 6 1 3 5

> x <- c(1.2, 4.2, 4.5, -5.6, 6.5, 1.2)
> rank(x, ties.method = "first")

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

- **Note:** Solo per `ties.method = "average"` e `ties.method = "first"` la somma del vettore finale rimane uguale a $n\,(n+1)\,/\,2$.

1.15 Funzioni di troncamento e di arrotondamento

trunc()

- **Package:** base
- **Input:**

  x valore numerico

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

$$[\, x \,]$$

- **Example:**

```
> trunc(x = 2)

[1] 2

> trunc(x = 2.999)

[1] 2

> trunc(x = -2.01)

[1] -2
```

floor()

- **Package:** base
- **Input:**

  x valore numerico

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

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

- **Example:**

```
> floor(x = 2)

[1] 2

> floor(x = 2.99)

[1] 2

> floor(x = -2.01)

[1] -3
```

ceiling()

- **Package:** base
- **Input:**

  x valore numerico

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

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

- **Example:**

```
> ceiling(x = 2)

[1] 2

> ceiling(x = 2.001)

[1] 3

> ceiling(x = -2.01)

[1] -2
```

round()

- **Package:** base
- **Input:**

  x valore numerico

  digits valore naturale n

- **Description:** arrotonda al numero di cifre specificato da n
- **Example:**

```
> 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
- **Input:**

  `x` valore numerico

  `digits` valore naturale n
- **Description:** arrotonda al numero di cifre significative specificate da n
- **Example:**

```
> 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
- **Input:**

  `x` oggetto numerico
- **Description:** trasforma un valore decimale in frazionario
- **Example:**

```
> 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 = FALSE)
> 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
- **Input:**

  x oggetto numerico

- **Description:** approssimazione razionale
- **Example:**

```
> matrice <- matrix(data = c(1.2, 34, 4.3, 4.2), nrow = 2, ncol = 2,
+     byrow = FALSE)
> matrice

     [,1] [,2]
[1,]  1.2  4.3
[2,] 34.0  4.2

> det(matrice)

[1] -141.16

> solve(matrice) %*% matrice

             [,1]          [,2]
[1,] 1.000000e+00 -2.303930e-17
[2,] 2.428613e-17  1.000000e+00

> rational(x = solve(matrice) %*% matrice)

     [,1] [,2]
[1,]    1    0
[2,]    0    1
```

1.16 Funzioni avanzate

gamma()

- **Package:** base
- **Input:**

  x valore numerico tale che $x > 0$

- **Description:** funzione gamma
- **Formula:**

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

- **Example:**

```
> gamma(x = 3.45)

[1] 3.146312

> gamma(x = 5)

[1] 24
```

lgamma()

- **Package:** base
- **Input:**

  x valore numerico tale che $x > 0$

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

$$\log\left(\Gamma(x)\right)$$

- **Example:**

```
> 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
- **Input:**

  x valore numerico tale che $x > 0$

- **Description:** funzione digamma
- **Formula:**

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

- **Example:**

```
> digamma(x = 2.45)

[1] 0.6783387

> digamma(x = 5.3)

[1] 1.570411
```

trigamma()

- **Package:** base
- **Input:**

  x valore numerico tale che $x > 0$

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

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

- **Example:**

```
> trigamma(x = 2.45)

[1] 0.5024545

> trigamma(x = 5.3)

[1] 0.2075909
```

psigamma()

- **Package:** base
- **Input:**

  x valore numerico tale che $x > 0$

  deriv valore naturale n

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

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

- **Example:**

```
> psigamma(x = 2.45, deriv = 0)

[1] 0.6783387

> digamma(x = 2.45)

[1] 0.6783387

> psigamma(x = 5.3, deriv = 1)

[1] 0.2075909

> trigamma(x = 5.3)

[1] 0.2075909
```

beta()

- **Package:** base
- **Input:**

  a valore numerico tale che $a > 0$

  b valore numerico tale che $b > 0$

- **Description:** funzione beta
- **Formula:**

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

- **Example:**

```
> 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
- **Input:**

  a valore numerico tale che $a > 0$

  b valore numerico tale che $b > 0$

- **Description:** logaritmo naturale della funzione beta
- **Formula:**

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

- **Example:**

```
> 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
```

fbeta()

- **Package:** MASS
- **Input:**

  `x` valore numerico tale che $x > 0$ e $x < 1$

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

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

- **Description:** funzione beta
- **Formula:**

$$x^{a-1}\,(1-x)^{b-1}$$

- **Example:**

```
> x <- 0.67
> a <- 3.45
> b <- 2.3
> x^(a - 1) * (1 - x)^(b - 1)
```

```
[1] 0.08870567
```

```
> fbeta(x = 0.67, a = 3.45, b = 2.3)
```

```
[1] 0.08870567
```

```
> x <- 0.12
> a <- 5
> b <- 4
> x^(a - 1) * (1 - x)^(b - 1)
```

```
[1] 0.0001413100
```

```
> fbeta(x = 0.12, a = 5, b = 4)
```

```
[1] 0.0001413100
```

sigmoid()

- **Package:** e1071
- **Input:**

  x valore numerico

- **Description:** funzione sigmoide
- **Formula:**

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

- **Example:**

```
> 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
- **Input:**

  x valore numerico

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

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

- **Example:**

```
> 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
- **Input:**

  x valore numerico

- **Description:** derivata seconda della funzione sigmoide
- **Formula:**

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

- **Example:**

```
> 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
```

besselI()

- **Package:** base
- **Input:**

  x valore numerico tale che $x > 0$

  nu valore naturale

- **Description:** funzione BesselI
- **Example:**

```
> besselI(x = 2.3, nu = 3)

[1] 0.3492232

> besselI(x = 1.6, nu = 2)

[1] 0.3939673
```

besselJ()

- **Package:** base
- **Input:**

  x valore numerico tale che $x > 0$
  nu valore naturale

- **Description:** funzione BesselJ
- **Example:**

```
> besselJ(x = 2.3, nu = 3)

[1] 0.1799789

> besselJ(x = 1.6, nu = 2)

[1] 0.2569678
```

besselK()

- **Package:** base
- **Input:**

  x valore numerico tale che $x > 0$
  nu valore naturale

- **Description:** funzione BesselK
- **Example:**

```
> besselK(x = 2.3, nu = 3)

[1] 0.3762579

> besselK(x = 1.6, nu = 2)

[1] 0.4887471
```

besselY()

- **Package:** base
- **Input:**

  x valore numerico tale che $x > 0$
  nu valore naturale

- **Description:** funzione BesselY
- **Example:**

```
> besselY(x = 2.3, nu = 3)

[1] -0.8742197

> besselY(x = 1.6, nu = 2)

[1] -0.8548994
```

1.17 Funzioni sui numeri complessi

complex()

- **Package:** base
- **Input:**

  `real` parte reale α

  `imaginary` parte immaginaria β

  `modulus` modulo r

  `argument` argomento ϕ
- **Description:** numero complesso
- **Formula:**

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

- **Example:**

```
> complex(real = 1, imaginary = 3)

[1] 1+3i

> complex(modulus = Mod(1 + 3i), argument = Arg(1 + 3i))

[1] 1+3i

> complex(real = -3, imaginary = 4)

[1] -3+4i

> complex(modulus = Mod(-3 + 4i), argument = Arg(-3 + 4i))

[1] -3+4i
```

Re()

- **Package:** base
- **Input:**

  `x` numero complesso
- **Description:** parte reale
- **Formula:**

$$\alpha$$

- **Example:**

```
> Re(x = 2 + 3i)

[1] 2

> Re(x = -3 + 4i)

[1] -3
```

Im()

- **Package:** base
- **Input:**

  x numero complesso

- **Description:** parte immaginaria
- **Formula:**

$$\beta$$

- **Example:**

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

[1] 3

> Im(x = 3 - 4i)

[1] -4
```

Mod()

- **Package:** base
- **Input:**

  x numero complesso

- **Description:** modulo
- **Formula:**

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

- **Example:**

```
> x <- 2 + 3i
> sqrt(2^2 + 3^2)

[1] 3.605551

> Mod(x = 2 + 3i)

[1] 3.605551

> x <- -3 + 4i
> sqrt((-3)^2 + 4^2)

[1] 5

> Mod(x = -3 + 4i)

[1] 5

> x <- 3 + 4i
> sqrt(3^2 + 4^2)

[1] 5

> Mod(x = 3 + 4i)
```

```
[1] 5

> abs(x = 3 + 4i)

[1] 5
```

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

Arg()

- **Package:** `base`
- **Input:**

  x numero complesso
- **Description:** argomento
- **Formula:**

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

- **Example:**

```
> x <- 2 + 3i
> atan(3/2)

[1] 0.9827937

> Arg(x = 2 + 3i)

[1] 0.9827937

> x <- 4 + 5i
> atan(5/4)

[1] 0.8960554

> Arg(x = 4 + 5i)

[1] 0.8960554
```

Conj()

- **Package:** `base`
- **Input:**

  x numero complesso
- **Description:** coniugato
- **Formula:**

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

- **Example:**

```
> Conj(x = 2 + 3i)

[1] 2-3i

> Conj(x = -3 + 4i)

[1] -3-4i
```

is.real()

- **Package:** base
- **Input:**

  x valore numerico

- **Description:** segnalazione di valore numerico reale
- **Example:**

```
> is.real(x = 2 + 3i)

[1] FALSE

> is.real(x = 4)

[1] TRUE
```

is.complex()

- **Package:** base
- **Input:**

  x valore numerico

- **Description:** segnalazione di valore numerico complesso
- **Example:**

```
> is.complex(x = 2 + 3i)

[1] TRUE

> is.complex(x = 4)

[1] FALSE
```

1.18 Funzioni cumulate

cumsum()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

- **Description:** somma cumulata
- **Formula:**

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

- **Example:**

```
> x <- c(1, 2, 4, 3, 5, 6)
> cumsum(x)

[1]  1  3  7 10 15 21
```

```
> x <- c(1, 2.3, 4.5, 6.7, 2.1)
> cumsum(x)

[1]  1.0  3.3  7.8 14.5 16.6
```

cumprod()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

- **Description:** prodotto cumulato
- **Formula:**

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

- **Example:**

```
> x <- c(1, 2, 4, 3, 5, 6)
> cumprod(x)

[1]   1   2   8  24 120 720

> x <- c(1, 2.3, 4.5, 6.7, 2.1)
> cumprod(x)

[1]   1.0000   2.3000  10.3500  69.3450 145.6245
```

cummin()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

- **Description:** minimo cumulato
- **Formula:**

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

- **Example:**

```
> x <- c(3, 4, 3, 2, 4, 1)
> cummin(x)

[1] 3 3 3 2 2 1

> x <- c(1, 3, 2, 4, 5, 1)
> cummin(x)

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

cummax()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n
- **Description:** massimo cumulato
- **Formula:**

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

- **Example:**

```
> x <- c(1, 3, 2, 4, 5, 1)
> cummax(x)

[1] 1 3 3 4 5 5

> x <- c(1, 3, 2, 4, 5, 1)
> cummax(x)

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

1.19 Funzioni in parallelo

pmin()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

  y vettore numerico di dimensione n
- **Description:** minimo in parallelo
- **Formula:**

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

- **Example:**

```
> x <- c(1.2, 2.3, 0.11, 4.5)
> y <- c(1.1, 2.1, 1.3, 4.4)
> pmin(x, y)

[1] 1.10 2.10 0.11 4.40

> x <- c(1.2, 2.3, 0.11, 4.5)
> y <- c(1.1, 2.1, 1.1, 2.1)
> pmin(x, y)

[1] 1.10 2.10 0.11 2.10
```

pmax()

- **Package:** base
- **Input:**

  `x` vettore numerico di dimensione n

  `y` vettore numerico di dimensione n

- **Description:** massimo in parallelo
- **Formula:**

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

- **Example:**

```
> x <- c(1.2, 2.3, 0.11, 4.5)
> y <- c(1.1, 2.1, 1.3, 4.4)
> pmax(x, y)

[1] 1.2 2.3 1.3 4.5

> x <- c(1.2, 2.3, 0.11, 4.5)
> y <- c(1.1, 2.1, 1.1, 2.1)
> pmax(x, y)

[1] 1.2 2.3 1.1 4.5
```

1.20 Funzioni di analisi numerica

optimize()

- **Package:** stats
- **Input:**

  `f` funzione $f(x)$

  `lower` estremo inferiore

  `upper` estremo superiore

  `maximum = TRUE / FALSE` massimo oppure minimo

  `tol` tolleranza

- **Description:** 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 = TRUE`

$$\max_x f(x)$$

`maximum = FALSE`

$$\min_x f(x)$$

- **Example:**

```
> f <- function(x) x * exp(-x^3) - (log(x))^2
> optimize(f, lower = 0.3, upper = 1.5, maximum = TRUE, tol = 1e-04)

$maximum
[1] 0.8374697

$objective
[1] 0.4339975

> f <- function(x) (x - 0.1)^2
> optimize(f, lower = 0, upper = 1, maximum = FALSE, tol = 1e-04)

$minimum
[1] 0.1

$objective
[1] 7.70372e-34

> f <- function(x) dchisq(x, df = 8)
> optimize(f, lower = 0, upper = 10, maximum = TRUE, tol = 1e-04)

$maximum
[1] 5.999999

$objective
[1] 0.1120209
```

optim()

- **Package:** stats
- **Input:**

  `par` valore di partenza

  `fn` funzione $f(x)$

  `method = "Nelder-Mead" / "BFGS" / "CG" / "L-BFGS-B" / "SANN"` metodo di ottimizzazione
- **Description:** ottimizzazione
- **Output:**

  `par` punto di ottimo

  `value` valore assunto dalla funzione nel punto individuato
- **Example:**

```
> f <- function(x) x * exp(-x^3) - (log(x))^2
> optim(par = 1, fn = f, method = "BFGS")$par

[1] 20804.91

> optim(par = 1, fn = f, method = "BFGS")$value

[1] -98.86214

> f <- function(x) (x - 0.1)^2
> optim(par = 1, fn = f, method = "BFGS")$par

[1] 0.1
```

```
> optim(par = 1, fn = f, method = "BFGS")$value

[1] 7.70372e-34

> f <- function(x) dchisq(x, df = 8)
> optim(par = 1, fn = f, method = "BFGS")$par

[1] 0.0003649698

> optim(par = 1, fn = f, method = "BFGS")$value

[1] 5.063142e-13

> nLL <- function(mu, x) {
+     z <- mu * x
+     lz <- log(z)
+     L1 <- sum(lz)
+     L2 <- mu/2
+     LL <- -(L1 - L2)
+     LL
+ }
> x <- c(1.2, 3.4, 5.6, 6.1, 7.8, 8.6, 10.7, 12, 13.7, 14.7)
> optim(par = 10000, fn = nLL, method = "CG", x = x)$par

[1] 9950.6

> optim(par = 10000, fn = nLL, method = "CG", x = x)$value

[1] 4863.693
```

uniroot()

- **Package:** stats
- **Input:**

  f funzione $f(x)$

  lower estremo inferiore

  upper estremo superiore

  tol tolleranza

  maxiter mumero massimo di iterazioni

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

  root radice

  f.root valore assunto dalla funzione nel punto individuato

  iter numero di iterazioni

  estim.prec tolleranza

- **Formula:**

$$f(x) = 0$$

- **Example:**

```
> f <- function(x) exp(-x) - x
> uniroot(f, lower = 0, upper = 1, tol = 1e-04, maxiter = 1000)
```

```
$root
[1] 0.5671439

$f.root
[1] -9.448109e-07

$iter
[1] 3

$estim.prec
[1] 7.425e-05
```

```
> f <- function(x) log10(x) + x
> uniroot(f, lower = 0.1, upper = 1, tol = 1e-04, maxiter = 1000)
```

```
$root
[1] 0.3990136

$f.root
[1] 1.279136e-06

$iter
[1] 5

$estim.prec
[1] 5e-05
```

polyroot()

- **Package:** stats
- **Input:**

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

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

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

- **Example:**

```
> k <- 3
> a1 <- 3
> a2 <- -2
> a3 <- 2
> a <- c(a1, a2, a3)
> polyroot(a)
```

```
[1] 0.5+1.118034i 0.5-1.118034i
```

```
> radice1 <- 0.5 + (0+1.118034i)
> a1 + a2 * radice1 + a3 * radice1^2
```

```
[1] -5.0312e-08+0i
```

```
> radice2 <- 0.5 - (0+1.118034i)
> a1 + a2 * radice2 + a3 * radice2^2
```

```
[1] -5.0312e-08+0i
```

```
> k <- 4
> a1 <- 3
> a2 <- -2
> a3 <- 2
> a4 <- -1
> a <- c(a1, a2, a3, a4)
> polyroot(a)
```

```
[1] 0.094732+1.283742i 0.094732-1.283742i 1.810536+0.000000i
```

```
> radice1 <- 0.09473214 + (0+1.283742i)
> a1 + a2 * radice1 + a3 * radice1^2 + a4 * radice1^3
```

```
[1] 7.477461e-07-5.808714e-07i
```

```
> radice2 <- 0.09473214 - (0+1.283742i)
> a1 + a2 * radice2 + a3 * radice2^2 + a4 * radice2^3
```

```
[1] 7.477461e-07+5.808714e-07i
```

```
> radice3 <- 1.81053571 + (0+0i)
> a1 + a2 * radice3 + a3 * radice3^2 + a4 * radice3^3
```

```
[1] 1.729401e-08+0i
```

D()

- **Package:** stats
- **Input:**

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

  `name` variabile x di derivazione

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

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

- **Example:**

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

```
-(exp(-x) + 1)
```

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

```
exp(-a)
```

DD()

- **Package:**
- **Input:**

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

  `name` variabile x di derivazione

  `order` il valore k dell'ordine di derivazione

- **Description:** derivata simbolica al k-esimo ordine
- **Formula:**

$$\frac{d^k}{d^k x}\, f(x)$$

- **Example:**

```
> DD(expr = expression(exp(-x) - x), name = "x", order = 1)
> DD(expr = expression(x * exp(-a)), name = "a", order = 2)
```

integrate()

- **Package:** `stats`
- **Input:**

  `f` funzione $f(x)$

  `lower` estremo inferiore a di integrazione

  `upper` estremo superiore b di integrazione

  `subdivisions` mumero di suddivisioni dell'intervallo di integrazione

- **Description:** integrazione numerica
- **Output:**

  `value` integrale definito

- **Formula:**

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

- **Example:**

```
> 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

> f <- function(x) dnorm(x)
> integrate(f, lower = -1.96, upper = 1.96, subdivisions = 150)

0.9500042 with absolute error < 1.0e-11
```

1.21 Costanti

pi

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

$$\pi$$

- **Example:**

```
> pi
[1] 3.141593
> 2 * pi
[1] 6.283185
```

Inf

- **Package:**
- **Description:** infinito
- **Formula:**

$$\pm \infty$$

- **Example:**

```
> 2/0
[1] Inf
> -2/0
[1] -Inf
> 0^Inf
[1] 0
> exp(-Inf)
[1] 0
> 0/Inf
[1] 0
> Inf - Inf
[1] NaN
> Inf/Inf
[1] NaN
> exp(Inf)
[1] Inf
```

NaN

- **Package:**
- **Description:** not a number
- **Example:**

```
> Inf - Inf

[1] NaN

> 0/0

[1] NaN
```

NA

- **Package:**
- **Description:** not available
- **Example:**

```
> x <- c(1.2, 3.4, 5.6, NA)
> mean(x)

[1] NA

> mean(x, na.rm = TRUE)

[1] 3.4
```

NULL

- **Package:**
- **Description:** oggetto nullo
- **Example:**

```
> x <- c(1.2, 3.4, 5.6)
> names(x) <- c("a", "b", "c")
> names(x) <- NULL
> x

[1] 1.2 3.4 5.6
```

TRUE

- **Package:**
- **Description:** vero
- **Example:**

```
> TRUE | TRUE

[1] TRUE

> TRUE & TRUE

[1] TRUE
```

T

- **Package:** base
- **Description:** vero
- **Example:**

```
> T
[1] TRUE
> T & T
[1] TRUE
```

FALSE

- **Package:**
- **Description:** falso
- **Example:**

```
> FALSE | TRUE
[1] TRUE
> FALSE & TRUE
[1] FALSE
```

F

- **Package:** base
- **Description:** falso
- **Example:**

```
> F
[1] FALSE
> F | T
[1] TRUE
```

1.22 Miscellaneous

list()

- **Package:** base
- **Description:** creazione di un oggetto lista
- **Example:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1)
> y <- c(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 <- c(1, 2.3, 4.5, 6.7, 8.9)
> y <- c(154, 109, 137, 115, 140)
> z <- c(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  92 146

> 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 <- c(1, 2, 3)
> y <- c(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 <- c(1, 2, 3)
> y <- c(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
- **Input:**

  x oggetto lista

  FUN funzione
- **Description:** applica la funzione FUN ad ogni elemento di lista
- **Example:**

```
> vec1 <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1)
> mean(vec1)

[1] 7.0125

> vec2 <- c(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 <- c(1, 2.3, 4.5, 6.7, 8.9)
> sd(vec1)

[1] 3.206556

> vec2 <- c(154, 109, 137, 115, 140)
> sd(vec2)

[1] 18.61451

> vec3 <- c(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
```

.Last.value

- **Package:** base
- **Description:** ultimo valore calcolato
- **Example:**

```
> 2 + 4

[1] 6

> .Last.value

[1] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
[7] "base"

> 3 * 4^4.2

[1] 1013.382

> .Last.value

[1] "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"
[7] "base"
```

identical()

- **Package:** base
- **Description:** uguaglianza tra due oggetti
- **Example:**

```
> u <- c(1, 2, 3)
> v <- c(1, 2, 4)
> if (identical(u, v)) print("uguali") else print("non uguali")

[1] "non uguali"

> u <- c(1, 2, 3)
> v <- c(1, 3, 2)
> identical(u, v)

[1] FALSE
```

any()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

- **Description:** restituisce TRUE se almeno un elemento del vettore soddisfa ad una condizione fissata
- **Example:**

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

[1] FALSE FALSE FALSE FALSE FALSE  TRUE

> any(x < 2)

[1] TRUE

> x <- c(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
- **Input:**

  x vettore numerico di dimensione n
- **Description:** restituisce TRUE se tutti gli elementi del vettore soddisfano ad una condizione fissata
- **Example:**

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

[1] FALSE FALSE FALSE FALSE FALSE  TRUE

> all(x < 2)

[1] FALSE

> x <- c(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
- **Input:**

  x vettore numerico di dimensione n

  table vettore numerico y di dimensione m

  nomatch alternativa da inserire al posto di NA
- **Description:** per ogni elemento di x restituisce la posizione della prima occorrenza in y
- **Example:**

```
> x <- c(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 <- c(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
- **Input:**

  `X` vettore numerico x di dimensione n

  `Y` vettore numerico y di dimensione m

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

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

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

- **Example:**

```
> 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
- **Input:**

  `x` oggetto

- **Description:** crea una espressione simbolica
- **Example:**

```
> u <- c(4.3, 5.5, 6.8, 8)
> w <- c(4, 5, 6, 7)
> z <- expression(x = u/w)
> z

expression(x = u/w)

> u <- c(1.2, 3.4, 4.5)
> w <- c(1, 2, 44)
> z <- expression(x = u * w)
> z

expression(x = u * w)
```

eval()

- **Package:** base
- **Input:**

  `expr` espressione simbolica

- **Description:** valuta una espressione simbolica
- **Example:**

```
> u <- c(4.3, 5.5, 6.8, 8)
> w <- c(4, 5, 6, 7)
> z <- expression(x = u/w)
> eval(expr = z)

[1] 1.075000 1.100000 1.133333 1.142857

> u <- c(1.2, 3.4, 4.5)
> w <- c(1, 2, 44)
> z <- expression(expr = u * w)
> eval(z)

[1]   1.2   6.8 198.0
```

replace()

- **Package:** base
- **Input:**

  `x` vettore numerico di dimensione n

  `list` indice dell'elemento da rimpiazzare

  `values` valore da inserire

- **Description:** rimpiazza un elemento del vettore x
- **Example 1:**

```
> x <- c(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
```

- **Example 2:**

```
> x <- c(1.2, 3.4, 5.6, 7.8)
> replace(x, list = 3, values = 8.9)

[1] 1.2 3.4 8.9 7.8

> x

[1] 1.2 3.4 5.6 7.8
```

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

e

- **Package:** base
- **Description:** scrittura rapida di un valore numerico potenza di 10
- **Example:**

```
> 1e3
[1] 1000
> -2e-2
[1] -0.02
> 1e-2
[1] 0.01
> 3e4
[1] 30000
```

even()

- **Package:** gtools
- **Input:**

  x valore naturale
- **Description:** verifica numero pari
- **Example:**

```
> even(x = 22)
[1] TRUE
> even(x = 7)
[1] FALSE
```

odd()

- **Package:** gtools
- **Input:**

  x valore naturale
- **Description:** verifica numero dispari
- **Example:**

```
> odd(x = 22)
[1] FALSE
> odd(x = 7)
[1] TRUE
```

'

- **Package:** base
- **Description:** notazione polacca inversa (RPN)
- **Example:**

```
> 1 + 2

[1] 3

> 3 * 4.2

[1] 12.6
```

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

gcd()

- **Package:** schoolmath
- **Input:**

  x valore naturale

  y valore naturale

- **Description:** massimo comun divisore
- **Example:**

```
> gcd(x = 6, y = 26)

[1] 2

> gcd(x = 8, y = 36)

[1] 4
```

scm()

- **Package:** schoolmath
- **Input:**

  x valore naturale

  y valore naturale

- **Description:** minimo comune multiplo
- **Example:**

```
> scm(6, 14)

[1] 42

> scm(12, 16)

[1] 48
```

is.vector()

- **Package:** base
- **Input:**

  x oggetto

- **Description:** oggetto di tipo vettore
- **Example 1:**

```
> x <- c(1.2, 2.34, 4.5, 6.7, 8.9)
> is.vector(x)

[1] TRUE

> is.matrix(x)

[1] FALSE
```

- **Example 2:**

```
> x <- matrix(data = 1:12, nrow = 3, ncol = 4)
> x

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

> is.vector(x)

[1] FALSE

> is.matrix(x)

[1] TRUE
```

- **Example 3:**

```
> x <- matrix(data = 1:12, nrow = 3, ncol = 4)
> x

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

> is.vector(x)

[1] FALSE

> is.matrix(x)

[1] TRUE
```

is.matrix()

- **Package:** base
- **Input:**

  x oggetto

- **Description:** oggetto di tipo matrice
- **Example 1:**

```
> x <- c(1.2, 2.34, 4.5, 6.7, 8.9)
> is.vector(x)

[1] TRUE

> is.matrix(x)

[1] FALSE
```

- **Example 2:**

```
> x <- matrix(data = 1:12, nrow = 3, ncol = 4)
> x

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

> is.vector(x)

[1] FALSE

> is.matrix(x)

[1] TRUE
```

- **Example 3:**

```
> x <- matrix(data = 1:12, nrow = 3, ncol = 4)
> x

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

> is.vector(x)

[1] FALSE

> is.matrix(x)

[1] TRUE
```

Capitolo 2

Vettori, Matrici ed Arrays

2.1 Creazione di Vettori

c()

- **Package:** base
- **Input:**

  ... oggetti da concatenare

  recursive = TRUE / FALSE concatenazione per oggetti di tipo list()

- **Description:** funzione di concatenazione
- **Example:**

```
> 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"
```

```
> x <- c(x, 2)
> x
```

```
[1] "a" "1" "2"
```

```
> lista <- list(primo = c(1, 2, 3), secondo = c(1.2, 5.6))
> lista
```

```
$primo
[1] 1 2 3

$secondo
[1] 1.2 5.6
```

```
> vettore <- c(lista, recursive = TRUE)
> vettore
```

```
  primo1   primo2   primo3 secondo1 secondo2 
     1.0      2.0      3.0      1.2      5.6 
```

```
> y <- 1.2
> z <- y[-1]
> z
```

```
numeric(0)
```

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

scan()

- **Package:** base
- **Input:**

  `what = double(0) / "character"` tipo dei dati numerico oppure carattere

- **Description:** creazione di un vettore
- **Example:**

```
> x <- scan(what = double(0))
> x <- scan(what = "character")
```

[ ]

- **Package:** base
- **Input:**

  `x` vettore alfanumerico di dimensione n

- **Description:** estrazione di elementi da un vettore
- **Example:**

```
> x <- c(1.2, 3.4, 5.6, 7.8, 9, 9.9)
> x

[1] 1.2 3.4 5.6 7.8 9.0 9.9

> x[2]

[1] 3.4

> x[c(1, 3, 4)]

[1] 1.2 5.6 7.8

> x[1:3]

[1] 1.2 3.4 5.6

> x[-c(1:3)]

[1] 7.8 9.0 9.9

> x[-(1:3)]

[1] 7.8 9.0 9.9

> x[x %in% c(1.2, 7.8)]

[1] 1.2 7.8

> x[x > 6.3]

[1] 7.8 9.0 9.9

> x[x > 6.3 & x < 9.7]
```

```
[1] 7.8 9.0

> x[c(TRUE, TRUE, FALSE, FALSE, TRUE, TRUE)]

[1] 1.2 3.4 9.0 9.9

> x[7]

[1] NA

> x[0]

numeric(0)

> x[c(1, 2, NA)]

[1] 1.2 3.4  NA

> 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

> x["a"]

  a
1.2
```

names()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

- **Description:** assegnazioni di nomi agli elementi di un vettore
- **Example:**

```
> x <- c(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

> names(x) <- NULL
> names(x)

NULL
```

vector()

- **Package:** base
- **Input:**

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

  `length` valore n della dimensione

- **Description:** inizializzazione di un vettore di dimensione n
- **Example:**

```
> 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
- **Input:**

  `length` dimensione

- **Description:** inizializzazione di un vettore numerico di dimensione n
- **Example:**

```
> x <- numeric(length = 5)
> x

[1] 0 0 0 0 0

> x <- numeric(length = 4)
> x

[1] 0 0 0 0
```

complex()

- **Package:** base
- **Input:**

  `length` dimensione

- **Description:** inizializzazione di un vettore complesso di dimensione n
- **Example:**

```
> 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
- **Input:**

  `length` dimensione

- **Description:** inizializzazione di un vettore logico di dimensione n
- **Example:**

```
> x <- logical(length = 5)
> x

[1] FALSE FALSE FALSE FALSE FALSE

> x <- logical(length = 4)
> x

[1] FALSE FALSE FALSE FALSE
```

head()

- **Package:** utils
- **Input:**

  `x` vettore numerico di dimensione m

  `n` numero di elementi

- **Description:** seleziona i primi n elementi
- **Example:**

```
> x <- c(1.2, 3.2, 3.3, 2.5, 5, 5.6)
> head(x, n = 2)

[1] 1.2 3.2

> x <- c(4.5, 6.7, 8.9, 7.7, 11.2)
> head(x, n = 3)

[1] 4.5 6.7 8.9
```

tail()

- **Package:** utils
- **Input:**

  x vettore numerico di dimensione m

  n numero di elementi
- **Description:** seleziona gli ultimi n elementi
- **Example:**

```
> x <- c(1.2, 3.2, 3.3, 2.5, 5, 5.6)
> tail(x, n = 3)

[1] 2.5 5.0 5.6

> x <- c(4.5, 6.7, 8.9, 7.7, 11.2)
> tail(x, n = 2)

[1]  7.7 11.2
```

%o%

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

  y vettore numerico di dimensione m
- **Description:** prodotto esterno
- **Formula:**

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

- **Example:**

```
> x <- c(1, 2, 3, 4)
> n <- 4
> y <- c(1.2, 3.4)
> m <- 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 <- c(3, 4, 7)
> n <- 3
> y <- c(1.1, 2.2, 3.3)
> m <- 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
- **Input:**

  `x` vettore numerico di dimensione n

  `values` valore v numerico

  `after` valore j naturale

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

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

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

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

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

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

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

- **Example:**

```
> x <- c(1.2, 3.4, 5.6)
> append(x, values = 6, after = -2)

[1] 6.0 1.2 3.4 5.6

> x <- c(1.2, 3.4, 5.6)
> append(x, values = 6, after = 2)

[1] 1.2 3.4 6.0 5.6

> x <- c(1.2, 3.4, 5.6)
> append(x, values = 6, after = 7)

[1] 1.2 3.4 5.6 6.0
```

sapply()

- **Package:** base
- **Input:**

  `X` vettore numerico di dimensione n

  `FUN` funzione scelta

- **Description:** applica `FUN` ad ogni elemento del vettore X
- **Example:**

```
> 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 <- c(2, 4, 7, 3, 5, 2, 9, 0)
> X <- c(2, 4, 6)
> myfun <- function(x) which(a > x)
> sapply(X, FUN = myfun)

[[1]]
[1] 2 3 4 5 7

[[2]]
[1] 3 5 7

[[3]]
[1] 3 7

> x <- c(1.5, 6.4, 9.6, 8.8, 7.7, 2.2, 4.8)
> sapply(X = 1:5, FUN = function(i) sample(x, size = 3, replace = FALSE))

     [,1] [,2] [,3] [,4] [,5]
[1,]  9.6  8.8  2.2  1.5  7.7
[2,]  1.5  9.6  9.6  7.7  9.6
[3,]  8.8  6.4  7.7  9.6  6.4

> x <- matrix(data = c(2, 3, 4, 5, 5, 4, 1, 3, 4, 7, 6, 5, 12,
+     13, 4, 11, 21, 10, 9, 7), nrow = 4, ncol = 5)
> x

     [,1] [,2] [,3] [,4] [,5]
[1,]    2    5    4   12   21
[2,]    3    4    7   13   10
[3,]    4    1    6    4    9
[4,]    5    3    5   11    7

> fattore <- factor(c(1, 2, 2, 1), labels = letters[1:2])
> fattore

[1] a b b a
Levels: a b

> sapply(X = 1:ncol(x), FUN = function(i) tapply(x[, i], INDEX = fattore,
+     FUN = mean))

  [,1] [,2] [,3] [,4] [,5]
a  3.5  4.0  4.5 11.5 14.0
b  3.5  2.5  6.5  8.5  9.5

> myfun <- function(x) prod(1:x)
> sapply(X = 1:5, myfun)

[1]   1   2   6  24 120

> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> sumsq <- function(b, xv = x, yv = y) {
+     yhat <- 1.2 + b * xv
+     sum((yv - yhat)^2)
+ }
> b <- seq(0, 2, by = 0.05)
> sapply(X = b, FUN = sumsq)
```

```
 [1] 367.20560 339.53785 313.06340 287.78225 263.69440 240.79985 219.09860
 [8] 198.59065 179.27600 161.15465 144.22660 128.49185 113.95040 100.60225
[15]  88.44740  77.48585  67.71760  59.14265  51.76100  45.57265  40.57760
[22]  36.77585  34.16740  32.75225  32.53040  33.50185  35.66660  39.02465
[29]  43.57600  49.32065  56.25860  64.38985  73.71440  84.23225  95.94340
[36] 108.84785 122.94560 138.23665 154.72100 172.39865 191.26960
```

subset()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

  subset selezione
- **Description:** sottoinsieme del vettore x
- **Example 1:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1, 6.7, 7.6, 6.8)
> subset(x, subset = x > 7.5)

[1] 7.8 7.6
```

- **Example 2:**

```
> x <- c(7.8, 6.6, 6.5, 6.6)
> subset(x, subset = x == 6.6)

[1] 6.6 6.6
```

2.2 Creazione di Matrici

matrix()

- **Package:** base
- **Input:**

  data vettore numerico di dimensione $n\,m$

  nrow numero n di righe

  ncol numero m di colonne

  byrow = TRUE / FALSE elementi disposti per riga oppure per colonna

  dimnames etichette di riga e di colonna
- **Description:** definizione di una matrice
- **Example:**

```
> n <- 2
> m <- 3
> x <- c(1, -0.2, 3, 1.1, -0.3, 3.2)
> A <- matrix(data = x, nrow = n, ncol = m, byrow = TRUE)
> 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(data = x, nrow = n, ncol = m, byrow = FALSE)
> 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(data = 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(data = 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(data = x, nrow = n, ncol = m, byrow = FALSE, dimnames = list(riga,
+     colonna))
> A

   c1 c2 c3
r1  1  4  7
r2  2  5  8
r3  3  6  9
```

dim()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione nm

- **Description:** dimensione
- **Example:**

```
> 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
```

rownames()

- **Package:** base
- **Input:**

  x matrice di dimensione $n \times m$

- **Description:** etichette di riga
- **Example:**

```
> x <- matrix(data = c(1, 3, 5, 2, 4, 1), nrow = 2, ncol = 3, byrow = TRUE)
> x
```

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

```
> rownames(x)
```

```
NULL
```

```
> rownames(x) <- c("r1", "r2")
> x
```

```
   [,1] [,2] [,3]
r1    1    3    5
r2    2    4    1
```

```
> rownames(x)
```

```
[1] "r1" "r2"
```

```
> x <- matrix(data = c(1, 4, 2, 3, 3, 2, 4, 1, 3.4, 4.3, 4.56,
+     11.1), nrow = 3, ncol = 4)
> x
```

```
     [,1] [,2] [,3]  [,4]
[1,]    1    3  4.0  4.30
[2,]    4    3  1.0  4.56
[3,]    2    2  3.4 11.10
```

```
> rownames(x)
```

```
NULL
```

```
> rownames(x) <- c("r1", "r2", "r3")
> x

   [,1] [,2] [,3]  [,4]
r1    1    3  4.0  4.30
r2    4    3  1.0  4.56
r3    2    2  3.4 11.10

> rownames(x)

[1] "r1" "r2" "r3"
```

colnames()

- **Package:** base
- **Input:**

  x matrice di dimensione $n \times m$

- **Description:** etichette di colonna
- **Example:**

```
> x <- matrix(data = c(1, 3, 5, 2, 4, 1), nrow = 2, ncol = 3, byrow = TRUE)
> 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"

> x <- matrix(data = c(1, 4, 2, 3, 3, 2, 4, 1, 3.4, 4.3, 4.56,
+     11.1), nrow = 3, ncol = 4)
> x

     [,1] [,2] [,3]  [,4]
[1,]    1    3  4.0  4.30
[2,]    4    3  1.0  4.56
[3,]    2    2  3.4 11.10

> colnames(x)

NULL
```

```
> colnames(x) <- c("c1", "c2", "c3", "c4")
> x

     c1 c2  c3    c4
[1,]  1  3 4.0  4.30
[2,]  4  3 1.0  4.56
[3,]  2  2 3.4 11.10

> colnames(x)

[1] "c1" "c2" "c3" "c4"
```

dimnames()

- **Package:** base
- **Input:**

  x matrice di dimensione $n \times m$

- **Description:** etichette di riga e di colonna
- **Example:**

```
> x <- matrix(data = 1:9, nrow = 3, ncol = 3)
> 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
- **Input:**

  A matrice di dimensione $n \times m$

- **Description:** estrazione di elementi da una matrice
- **Example:**

```
> A <- matrix(data = 1:9, nrow = 3, ncol = 3)
> dimnames(A) <- list(c("r1", "r2", "r3"), c("c1", "c2", "c3"))
> n <- 3
> m <- 3
> A[2, 3]

[1] 8

> A[1, ]

c1 c2 c3
 1  4  7

> A["r1", ]

c1 c2 c3
 1  4  7

> A[, 3]

r1 r2 r3
 7  8  9

> A[, "c3"]

r1 r2 r3
 7  8  9

> 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

> A[, c(2, 3)]

   c2 c3
r1  4  7
r2  5  8
r3  6  9

> A[, c("c2", "c3")]
```

```
   c2 c3
r1  4  7
r2  5  8
r3  6  9
```

```
> A[-1, ]
```

```
   c1 c2 c3
r2  2  5  8
r3  3  6  9
```

```
> A[, -3]
```

```
   c1 c2
r1  1  4
r2  2  5
r3  3  6
```

```
> A[A[, "c2"] > 4.1, ]
```

```
   c1 c2 c3
r2  2  5  8
r3  3  6  9
```

```
> x[x > 3]
```

```
[1] 4 5 6 7 8 9
```

```
> A <- matrix(data = c(1.2, 3.4, 5.6, 7.8, 9.1), nrow = 1, ncol = 5)
> is.matrix(A)
```

```
[1] TRUE
```

```
> myvec <- A[1, ]
> is.vector(myvec)
```

```
[1] TRUE
```

```
> myvec2 <- A[, 1]
> is.vector(myvec2)
```

```
[1] TRUE
```

```
> myvec3 <- A[1, , drop = FALSE]
> is.vector(myvec3)
```

```
[1] FALSE
```

```
> is.matrix(myvec3)
```

```
[1] TRUE
```

col()

- **Package:** base
- **Input:**

  data matrice di dimensione $n \times m$

- **Description:** colonna di appartenenza di ogni elemento
- **Example:**

```
> x <- matrix(data = 1:9, nrow = 3, ncol = 3)
> 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 <- matrix(data = c(1.1, 2.3, 4.5, 6.7, 8.8, 6.1), nrow = 2,
+     ncol = 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
- **Input:**

  data matrice di dimensione $n \times m$

- **Description:** riga di appartenenza di ogni elemento
- **Example:**

```
> x <- matrix(data = 1:9, nrow = 3, ncol = 3)
> 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 <- matrix(data = c(1.1, 2.3, 4.5, 6.7, 8.8, 6.1), nrow = 2,
+     ncol = 3)
> n <- 2
> m <- 3
> row(x)

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

head()

- **Package:** utils
- **Input:**

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

  `n` numero di righe
- **Description:** seleziona le prime n righe
- **Example:**

```
> x <- matrix(data = 1:9, nrow = 3, ncol = 3)
> 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 <- matrix(data = 1:9, nrow = 3, ncol = 3, byrow = TRUE)
> 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
- **Input:**

  data matrice di dimensione $k \times m$

  n numero di righe
- **Description:** seleziona le ultime n righe
- **Example:**

```
> x <- matrix(data = 1:9, nrow = 3, ncol = 3)
> 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 <- matrix(data = 1:9, nrow = 3, ncol = 3, byrow = TRUE)
> k <- 3
> m <- 3
> tail(x, n = 2)

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

vech()

- **Package:** fUtilities
- **Input:**

  x matrice di dimensione $m \times n$
- **Description:** seleziona gli elementi della sezione triangolare inferiore di una matrice simmetrica
- **Example:**

```
> x <- matrix(data = c(1, 2, 3, 4, 2, 4, 5, 6, 3, 5, 7, 8, 4, 6,
+     8, 9), nrow = , ncol = 4)
> 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] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    1    2    3    4    4    5    6    7    8     9
```

```
> x <- matrix(data = c(11, 12, 13, 12, 14, 15, 13, 15, 16), nrow = 3,
+     ncol = 3)
> x
```

```
     [,1] [,2] [,3]
[1,]   11   12   13
[2,]   12   14   15
[3,]   13   15   16
```

```
> vech(x)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]   11   12   13   14   15   16
```

xpnd()

- **Package:** MCMCpack
- **Input:**

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

  nrow numero n di righe

- **Description:** crea una matrice simmetrica a partire da un vettore
- **Example:**

```
> 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
- **Input:**

  A matrice di dimensione $n \times m$

- **Description:** numero di elementi
- **Formula:**

$$n\,m$$

- **Example:**

```
> A <- matrix(data = 1:9, nrow = 3, ncol = 3)
> 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 <- matrix(data = c(1.2, 4.5, 2.3, 3.1), nrow = 2, ncol = 2)
> 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
- **Input:**

  A matrice di dimensione $n \times m$

  B matrice di dimensione $n \times k$

- **Description:** unisce due matrici accostandole per colonna
- **Example:**

```
> A <- matrix(data = c(9.9, 1, 12), nrow = 3, ncol = 1)
> A

     [,1]
[1,]  9.9
[2,]  1.0
[3,] 12.0

> B <- matrix(data = 1:3, nrow = 3, ncol = 1)
> 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 <- matrix(data = 1:2, nrow = 2, ncol = 1)
> A
```

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

```
> B <- matrix(data = 3:4, nrow = 2, ncol = 1)
> 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
- **Input:**

  A matrice di dimensione $n \times m$

  B matrice di dimensione $k \times m$

- **Description:** unisce due matrici accostandole per riga
- **Example:**

```
> A <- matrix(data = c(9.9, 1, 12), nrow = 1, ncol = 3)
> A
```

```
     [,1] [,2] [,3]
[1,]  9.9    1   12
```

```
> B <- matrix(data = 1:3, nrow = 1, ncol = 3)
> 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 <- matrix(data = 1:2, nrow = 2, ncol = 1)
> A
```

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

```
> B <- matrix(data = 3:4, nrow = 2, ncol = 1)
> 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
- **Input:**

  data vettore numerico di dimensione n

- **Description:** matrice simmetrica di *Toeplitz* di dimensione $n \times n$
- **Example:**

```
> x <- 1:3
> n <- 3
> toeplitz(x)
```

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

```
> x <- c(-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 = FALSE)$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
```

hilbert()

- **Package:** fUtilities
- **Input:**

  n valore n naturale

- **Description:** matrice di *Hilbert*
- **Formula:**

$$1\,/\,(i+j-1)\quad\forall i\,=\,1,\,2,\,\ldots,\,n\quad\forall j\,=\,1,\,2,\,\ldots,\,n$$

- **Example:**

```
> n <- 5
> hilbert(n)
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 1.0000000 0.5000000 0.3333333 0.2500000 0.2000000
[2,] 0.5000000 0.3333333 0.2500000 0.2000000 0.1666667
[3,] 0.3333333 0.2500000 0.2000000 0.1666667 0.1428571
[4,] 0.2500000 0.2000000 0.1666667 0.1428571 0.1250000
[5,] 0.2000000 0.1666667 0.1428571 0.1250000 0.1111111
```

```
> n <- 7
> hilbert(n)
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]       [,6]       [,7]
[1,] 1.0000000 0.5000000 0.3333333 0.2500000 0.2000000 0.16666667 0.14285714
[2,] 0.5000000 0.3333333 0.2500000 0.2000000 0.1666667 0.14285714 0.12500000
[3,] 0.3333333 0.2500000 0.2000000 0.1666667 0.1428571 0.12500000 0.11111111
[4,] 0.2500000 0.2000000 0.1666667 0.1428571 0.1250000 0.11111111 0.10000000
[5,] 0.2000000 0.1666667 0.1428571 0.1250000 0.1111111 0.10000000 0.09090909
[6,] 0.1666667 0.1428571 0.1250000 0.1111111 0.1000000 0.09090909 0.08333333
[7,] 0.1428571 0.1250000 0.1111111 0.1000000 0.0909091 0.08333333 0.07692308
```

pascal()

- **Package:** fUtilities
- **Input:**

  n valore n naturale

- **Description:** matrice di *Pascal*
- **Example:**

```
> n <- 5
> pascal(n)

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    1    1    1    1
[2,]    1    2    3    4    5
[3,]    1    3    6   10   15
[4,]    1    4   10   20   35
[5,]    1    5   15   35   70

> n <- 7
> pascal(n)

     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    1    1    1    1    1    1    1
[2,]    1    2    3    4    5    6    7
[3,]    1    3    6   10   15   21   28
[4,]    1    4   10   20   35   56   84
[5,]    1    5   15   35   70  126  210
[6,]    1    6   21   56  126  252  462
[7,]    1    7   28   84  210  462  924
```

2.3 Operazioni sulle Matrici

rk()

- **Package:** fUtilities
- **Input:**

  A matrice di dimensione $n \times n$

- **Description:** rango cioé il numero di righe (colonne) linearmente indipendenti
- **Example:**

```
> A <- matrix(data = c(1, 4, 2, 8), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]    1    2
[2,]    4    8

> n <- 2
> rk(A)

[1] 1

> A <- matrix(data = c(1.2, 2.3, 4.5, 6.5, 7.6, 1.1, 2.3, 4.5,
+     6.7), nrow = 3, ncol = 3)
> 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
> rk(A)
```

```
[1] 3
```

det()

- **Package:** `base`
- **Input:**

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

- **Description:** determinante
- **Formula:**

$$\det(A)$$

- **Example:**

```
> A <- matrix(data = c(1, 4, -0.2, 5.6), nrow = 2, ncol = 2)
> A
```

```
     [,1] [,2]
[1,]    1 -0.2
[2,]    4  5.6
```

```
> n <- 2
> det(A)
```

```
[1] 6.4
```

```
> A <- matrix(data = c(1.2, 2.3, 4.5, 6.5, 7.6, 1.1, 2.3, 4.5,
+     6.7), nrow = 3, ncol = 3)
> 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
- **Input:**

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

  `logarithm = TRUE / FALSE` logaritmo naturale del modulo del determinante
- **Description:** determinante
- **Output:**

  `modulus` modulo

  `sign` segno
- **Formula:**

`logarithm = TRUE`

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

`sign`
$$\textbf{sign}\left(\det(A)\right)$$

`logarithm = FALSE`

`modulus`
$$|\det(A)|$$

`sign`
$$\textbf{sign}\left(\det(A)\right)$$

- **Example:**

```
> A <- matrix(data = c(1, 4, -0.2, 5.6), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]    1 -0.2
[2,]    4  5.6

> n <- 2
> abs(det(A))

[1] 6.4

> determinant(A, logarithm = FALSE)$modulus

[1] 6.4
attr(,"logarithm")
[1] FALSE

> sign(det(A))

[1] 1

> determinant(A, logarithm = FALSE)$sign

[1] 1

> A <- matrix(data = c(1.2, 4.5, 6.7, 8.9, 4.5, 6.6, 7.8, 7.5,
+     3.3), nrow = 3, ncol = 3)
> 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 = FALSE)$modulus

[1] 269.97
attr(,"logarithm")
[1] FALSE

> sign(det(A))

[1] 1

> determinant(A, logarithm = FALSE)$sign

[1] 1
```

determinant.matrix()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times n$

  logarithm = TRUE / FALSE logaritmo naturale del modulo del determinante
- **Description:** determinante
- **Output:**

  modulus modulo

  sign segno
- **Formula:**

  logarithm = TRUE

  modulus

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

  sign

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

  logarithm = FALSE

  modulus

  $$|\det(A)|$$

  sign

  $$\text{sign}\left(\det(A)\right)$$
- **Example:**

```
> A <- matrix(data = c(1, 4, -0.2, 5.6), nrow = 2, ncol = 2)
> A
```

```
     [,1] [,2]
[1,]    1 -0.2
[2,]    4  5.6

> n <- 2
> abs(det(A))

[1] 6.4

> determinant.matrix(A, logarithm = FALSE)$modulus

[1] 6.4
attr(,"logarithm")
[1] FALSE

> sign(det(A))

[1] 1

> determinant.matrix(A, logarithm = FALSE)$sign

[1] 1

> A <- matrix(data = c(1.2, 4.5, 6.7, 8.9, 4.5, 6.6, 7.8, 7.5,
+     3.3), nrow = 3, ncol = 3)
> 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 = FALSE)$modulus

[1] 269.97
attr(,"logarithm")
[1] FALSE

> sign(det(A))

[1] 1

> determinant.matrix(A, logarithm = FALSE)$sign

[1] 1
```

tr()

- **Package:** fUtilities
- **Input:**

  A matrice di dimensione $n \times n$

- **Description:** traccia
- **Formula:**

$$\sum_{i=1}^{n} a_{i,i}$$

- **Example:**

```
> A <- matrix(data = c(1, 4, 2, 8), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]    1    2
[2,]    4    8

> n <- 2
> tr(A)

[1] 9

> A <- matrix(data = c(1.2, 2.3, 4.5, 6.5, 7.6, 1.1, 2.3, 4.5,
+     6.7), nrow = 3, ncol = 3)
> 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
> tr(A)

[1] 15.5
```

norm()

- **Package:** fUtilities
- **Input:**

  A matrice di dimensione $n \times m$

  p = 1 / 2 / Inf massima somma assoluta di colonna, radice quadrata del massimo autovalore della matrice $A^T \, A$, massima somma assoluta di riga

- **Description:** norma
- **Formula:**

p = 1

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

`p = 2`

$$\max_i (\lambda_i) \quad \forall i = 1, 2, \ldots, m$$

`p = Inf`

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

- **Example:**

```
> n <- 2
> m <- 2
> A <- matrix(data = c(2.2, 3.4, 0.2, -1.2), nrow = 2, ncol = 2,
+      byrow = FALSE)
> A
```

```
     [,1] [,2]
[1,]  2.2  0.2
[2,]  3.4 -1.2
```

```
> max(abs(2.2) + abs(3.4), abs(0.2) + abs(-1.2))
```

```
[1] 5.6
```

```
> norm(A, p = 1)
```

```
[1] 5.6
```

```
> autovalori <- eigen(t(A) %*% A)$values
> sqrt(max(autovalori))
```

```
[1] 4.152189
```

```
> norm(A, p = 2)
```

```
[1] 4.152189
```

```
> max(abs(2.2) + abs(0.2), abs(3.4) + abs(-1.2))
```

```
[1] 4.6
```

```
> norm(A, p = Inf)
```

```
[1] 4.6
```

isPositiveDefinite()

- **Package:** fUtilities
- **Input:**

  x matrice di dimensione $n \times n$

- **Description:** matrice definita positiva
- **Example:**

```
> A <- matrix(data = c(1, 4, -0.2, 5.6), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]    1 -0.2
[2,]    4  5.6

> n <- 2
> isPositiveDefinite(A)

[1] TRUE

> A <- matrix(data = c(1.2, 2.3, 4.5, 6.5, 7.6, 1.1, 2.3, 4.5,
+     6.7), nrow = 3, ncol = 3)
> 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
> isPositiveDefinite(A)

[1] TRUE

> A <- matrix(data = c(-1, 1, 1, -1), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]   -1    1
[2,]    1   -1

> n <- 2
> isPositiveDefinite(A)

[1] FALSE
```

as.vector()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times m$

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

```
> A <- matrix(data = 1:9, nrow = 3, ncol = 3)
> 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 <- matrix(data = c(1.2, 2.3, 6.5, 7.6), nrow = 2, ncol = 2)
> 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
```

solve()

- **Package:** base
- **Input:**

  A matrice invertibile di dimensione $n \times n$

  B matrice di dimensione $n \times k$

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

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

- **Example:**

```
> A <- matrix(data = c(1, -0.2, 4, 5.6), nrow = 2, ncol = 2)
> 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 <- matrix(data = c(1, -0.2, 4, 5.6), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]  1.0  4.0
[2,] -0.2  5.6

> B <- c(11, -2)
> 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 <- matrix(data = c(1, -0.2, 4, 5.6), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]  1.0  4.0
[2,] -0.2  5.6

> B <- matrix(data = c(11, -2, 13, 4.1), nrow = 2, ncol = 2)
> 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
- **Input:**

  A matrice simmetrica di dimensione $n \times n$

  only.values = TRUE / FALSE calcola i soli autovalori

- **Description:** 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:**

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

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

- **Example:**

```
> A <- matrix(data = c(1.2, 3, 5.6, 3, 4, 6.7, 5.6, 6.7, 9.8),
+      nrow = 3, ncol = 3)
> 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
> 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 <- matrix(data = c(1.2, 2.3, 2.3, 2.2), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]  1.2  2.3
[2,]  2.3  2.2
```

```
> n <- 2
> 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
```

crossprod()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times m$

  B matrice di dimensione $n \times k$
- **Description:** prodotto scalare
- **Formula:**

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

- **Example:**

```
> A <- matrix(data = c(1.2, 3, 5.6, 3, 4, 6.7, 5.6, 6.7, 9.8),
+     nrow = 3, ncol = 3)
> 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 <- matrix(data = c(1.2, 3, 5.6, 3, 4, 6.7, 5.6, 6.7, 9.8),
+     nrow = 3, ncol = 3)
> 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 <- matrix(data = c(11, -2, 3.4, 4.1, 5, 7), nrow = 3, ncol = 2)
> 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
- **Input:**

  A matrice di dimensione $n \times m$

  B matrice di dimensione $k \times m$

- **Description:** prodotto scalare
- **Formula:**

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

- **Example:**

```
> A <- matrix(data = c(1.2, 3, 5.6, 3, 4, 6.7, 5.6, 6.7, 9.8),
+     nrow = 3, ncol = 3)
> 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 <- matrix(data = c(1.2, 3, 5.6, 3, 4, 6.7, 5.6, 6.7, 9.8),
+     nrow = 3, ncol = 3)
> 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 <- matrix(data = c(11, 4.1, -2, 5, 3.4, 7), nrow = 2, ncol = 3)
> 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
- **Input:**

  A matrice di dimensione $n \times m$

  B matrice di dimensione $n \times m$

- **Description:** prodotto di *Hadamard*
- **Formula:**

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

- **Example:**

```
> A <- matrix(data = 1:9, nrow = 3, ncol = 3)
> A

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

> B <- matrix(data = c(4.1, 2.3, 4.1, 5.4, 4.6, 4.2, 2.1, 3.2,
+     4.3), nrow = 3, ncol = 3)
> B

     [,1] [,2] [,3]
[1,]  4.1  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,]  4.1 21.6 14.7
[2,]  4.6 23.0 25.6
[3,] 12.3 25.2 38.7

> A <- matrix(data = c(1, 2, 3, 5), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]    1    3
[2,]    2    5

> B <- matrix(data = c(1.1, 2.3, 4.5, 6.7), nrow = 2, ncol = 2)
> 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
- **Input:**

  A matrice di dimensione $n \times m$

  B matrice di dimensione $m \times k$

- **Description:** prodotto scalare
- **Formula:**

$$A\,B$$

- **Example:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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 <- matrix(data = c(11, -1, 3.4, 4.1, 5, 7), nrow = 3, ncol = 2)
> B

     [,1] [,2]
[1,] 11.0  4.1
[2,] -1.0  5.0
[3,]  3.4  7.0

> n <- 3
> m <- 3
> k <- 2
> A %*% B

      [,1]   [,2]
[1,] 40.66  93.40
[2,] -4.40  34.18
[3,] 66.00 135.30

> A <- matrix(data = 1:2, nrow = 1, ncol = 2)
> A

     [,1] [,2]
[1,]    1    2

> B <- matrix(data = 3:4, nrow = 2, ncol = 1)
> B

     [,1]
[1,]    3
[2,]    4

> n <- 1
> m <- 2
> k <- 1
> A %*% B

     [,1]
[1,]   11
```

kronecker()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times m$

  B matrice di dimensione $h \times k$

- **Description:** prodotto di *Kronecker*
- **Formula:**

$$A \otimes B = \begin{pmatrix} a_{1,1} B & \cdots & a_{1,m} B \\ \vdots & \vdots & \vdots \\ a_{n,1} B & \cdots & a_{n,m} B \end{pmatrix}$$

- **Example:**

```
> A <- matrix(data = 1:3, nrow = 3, ncol = 1)
> A

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

> B <- matrix(data = 7:9, nrow = 1, ncol = 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 <- matrix(data = 1:2, nrow = 1, ncol = 2)
> A

     [,1] [,2]
[1,]    1    2

> B <- matrix(data = 3:4, nrow = 2, ncol = 1)
> 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
```

%x%

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times m$

  B matrice di dimensione $h \times k$

- **Description:** prodotto di *Kronecker*
- **Formula:**

$$A \otimes B = \begin{pmatrix} a_{1,1}\,B & \cdots & a_{1,m}\,B \\ \vdots & \vdots & \vdots \\ a_{n,1}\,B & \cdots & a_{n,m}\,B \end{pmatrix}$$

- **Example:**

```
> A <- matrix(data = 1:3, nrow = 3, ncol = 1)
> A

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

> B <- matrix(data = 7:9, nrow = 1, ncol = 3)
> B

     [,1] [,2] [,3]
[1,]    7    8    9

> n <- 3
> m <- 1
> h <- 1
> k <- 3
> A %x% B

     [,1] [,2] [,3]
[1,]    7    8    9
[2,]   14   16   18
[3,]   21   24   27

> A <- matrix(data = 1:2, nrow = 1, ncol = 2)
> A

     [,1] [,2]
[1,]    1    2

> B <- matrix(data = 3:4, nrow = 2, ncol = 1)
> B

     [,1]
[1,]    3
[2,]    4

> n <- 1
> m <- 2
> h <- 2
> k <- 1
> A %x% B

     [,1] [,2]
[1,]    3    6
[2,]    4    8
```

diag()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times n$

  x vettore numerico di dimensione n

  h valore naturale
- **Description:** estrae gli elementi diagonali o crea una matrice diagonale
- **Example:**

```
> A <- matrix(data = 1:9, nrow = 3, ncol = 3)
> 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
- **Input:**

  A matrice di dimensione $n \times m$
- **Description:** trasposta
- **Formula:**

$$A^T$$

- **Example:**

```
> A <- matrix(data = c(1.2, 3.4, 4.23, 1, 2, 3.4, 4.6, 7.8, 9.88),
+     nrow = 3, ncol = 3)
> 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 <- matrix(data = 1:2, nrow = 1, ncol = 2)
> A
```

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

```
> n <- 1
> m <- 2
> t(A)
```

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

aperm()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times m$

- **Description:** trasposta
- **Formula:**

$$A^T$$

- **Example:**

```
> A <- matrix(data = c(1.2, 3.4, 4.23, 1, 2, 3.4, 4.6, 7.8, 9.88),
+     nrow = 3, ncol = 3)
> 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 <- matrix(data = 1:2, nrow = 1, ncol = 2)
> A

     [,1] [,2]
[1,]    1    2

> n <- 1
> m <- 2
> t(A)

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

dim()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times m$

- **Description:** numero di righe e di colonne
- **Formula:**

$$n \quad m$$

- **Example:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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 <- matrix(data = c(1.2, 2.3, 6.5, 7.6), nrow = 2, ncol = 2)
> 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
- **Input:**

  A matrice di dimensione $n \times m$

- **Description:** numero di righe
- **Formula:**

$$n$$

- **Example:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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 <- matrix(data = c(1.2, 2.3, 6.5, 7.6), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]  1.2  6.5
[2,]  2.3  7.6

> nrow(A)

[1] 2
```

NROW()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times m$

- **Description:** numero di righe
- **Formula:**

$$n$$

- **Example:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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 <- matrix(data = c(1.2, 2.3, 6.5, 7.6), nrow = 2, ncol = 2)
> A
```

```
     [,1] [,2]
[1,]  1.2  6.5
[2,]  2.3  7.6
```

```
> NROW(A)
```

```
[1] 2
```

ncol()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times m$

- **Description:** numero di colonne
- **Formula:**

$$m$$

- **Example:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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 <- matrix(data = 1:2, nrow = 1, ncol = 2)
> A
```

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

```
> ncol(A)
```

```
[1] 2
```

NCOL()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times m$

- **Description:** numero di colonne
- **Formula:**

$$m$$

- **Example:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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 <- matrix(data = 1:2, nrow = 1, ncol = 2)
> A

     [,1] [,2]
[1,]    1    2

> NCOL(A)

[1] 2
```

rowSums()

- **Package:** fUtilities
- **Input:**

  A matrice di dimensione $n \times m$

- **Description:** somme di riga
- **Formula:**

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

- **Example:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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 <- matrix(data = c(1.2, 3.4, 4.5, 5.6), nrow = 2, ncol = 2)
> 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:** fUtilities
- **Input:**

  A matrice di dimensione $n \times m$
- **Description:** medie di riga
- **Formula:**

$$\frac{1}{m} \sum_{j=1}^{m} x_{ij} \quad \forall i = 1, 2, \ldots, n$$

- **Example:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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 <- matrix(data = c(1.2, 3.4, 4.5, 5.6), nrow = 2, ncol = 2)
> 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:** fUtilities
- **Input:**

  A matrice di dimensione $n \times m$

- **Description:** somme di colonna
- **Formula:**

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

- **Example:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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 <- matrix(data = c(1.2, 3.4, 4.5, 5.6), nrow = 2, ncol = 2)
> 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:** fUtilities
- **Input:**

  A matrice di dimensione $n \times m$

- **Description:** medie di colonna
- **Formula:**

$$\frac{1}{n} \sum_{i=1}^{n} x_{ij} \quad \forall j = 1, 2, \ldots, m$$

- **Example:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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 <- matrix(data = c(1.2, 3.4, 4.5, 5.6), nrow = 2, ncol = 2)
> 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
- **Input:**

  A matrice di dimensione $n \times m$

  group fattore f a k livelli di dimensione n

- **Description:** applica la funzione somma ad ogni gruppo di elementi in ciascuna colonna di A definito dai livelli di f
- **Example 1:**

```
> A <- matrix(data = c(1.2, 2.3, 4.3, 4.2, 4.2, 2.1, 2.2, 4), nrow = 4,
+     ncol = 2)
> 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 <- factor(rep(1:2, times = 2))
> k <- nlevels(f)
> k

[1] 2

> rowsum(A, f)

   [,1] [,2]
1   5.5  6.4
2   6.5  6.1
```

- **Example 2:**

```
> A <- matrix(data = c(1, 2, 3, 4, 7, 8, 9, 8), nrow = 4, ncol = 2)
> A

     [,1] [,2]
[1,]    1    7
[2,]    2    8
[3,]    3    9
[4,]    4    8

> n <- 4
> m <- 2
> k <- nlevels(f)
> k

[1] 2

> rowsum(A, f)

  [,1] [,2]
1    4   16
2    6   16
```

apply()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times m$

  MARGIN = 1 / 2 riga o colonna

  FUN funzione scelta

- **Description:** applica FUN ad ogni riga o colonna della matrice A
- **Example 1:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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
> apply(A, MARGIN = 1, FUN = mean)

[1] 4.966667 2.133333 7.600000
```

- **Example 2:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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
> apply(A, MARGIN = 2, FUN = mean)
```

```
[1] 1.266667 5.800000 7.633333
```

- **Example 3:**

```
> A <- matrix(data = c(2, -1, -10.2, 1, -1, 5, 5.8, 3, 1, 3, 3.1,
+     4), nrow = 4, ncol = 3)
> A
```

```
      [,1] [,2] [,3]
[1,]   2.0 -1.0  1.0
[2,]  -1.0  5.0  3.0
[3,] -10.2  5.8  3.1
[4,]   1.0  3.0  4.0
```

```
> n <- 4
> m <- 3
> apply(A, MARGIN = 2, FUN = sort)
```

```
      [,1] [,2] [,3]
[1,] -10.2 -1.0  1.0
[2,]  -1.0  3.0  3.0
[3,]   1.0  5.0  3.1
[4,]   2.0  5.8  4.0
```

- **Example 4:**

```
> A <- matrix(data = c(2, -1, -10.2, 1, -1, 5, 5.8, 3, 1, 3, 3.1,
+     4), nrow = 4, ncol = 3)
> A
```

```
      [,1] [,2] [,3]
[1,]   2.0 -1.0  1.0
[2,]  -1.0  5.0  3.0
[3,] -10.2  5.8  3.1
[4,]   1.0  3.0  4.0
```

```
> n <- 4
> m <- 3
> apply(A, MARGIN = 2, FUN = function(x) {
+     sort(x, decreasing = TRUE)
+ })
```

```
      [,1] [,2] [,3]
[1,]   2.0  5.8  4.0
[2,]   1.0  5.0  3.1
[3,]  -1.0  3.0  3.0
[4,] -10.2 -1.0  1.0
```

- **Example 5:**

```
> A <- matrix(data = c(1, 10, 100, 2, 20, 200, 3, 30, 300), nrow = 3,
+     ncol = 3)
> A
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]   10   20   30
[3,]  100  200  300
```

```
> n <- 3
> m <- 3
> apply(A, MARGIN = 2, FUN = cumsum)
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]   11   22   33
[3,]  111  222  333
```

```
> t(apply(A, MARGIN = 1, FUN = cumsum))
```

```
     [,1] [,2] [,3]
[1,]    1    3    6
[2,]   10   30   60
[3,]  100  300  600
```

solveCrossprod()

- **Package:** strucchange
- **Input:**

  A matrice di dimensione $n \times k$ di rango $k = \min\,(n,\, k)$

  method = qr / chol / solve algoritmo risolutivo

- **Description:** inversa del prodotto incrociato di X
- **Formula:**

$$(A^T\,A)^{-1}$$

- **Example 1:**

```
> A <- matrix(data = c(11, -2, 3.4, 4.1, 5, 7), nrow = 3, ncol = 2)
> A
```

```
     [,1] [,2]
[1,] 11.0  4.1
[2,] -2.0  5.0
[3,]  3.4  7.0
```

```
> n <- 3
> k <- 2
> 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
```

- **Example 2:**

```
> A <- matrix(data = c(1, 2, 3, 4, 7, 8, 9, 8), nrow = 4, ncol = 2)
> 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:** stats
- **Input:**

  `object` modello di regressione lineare con $k - 1$ variabili esplicative ed n unità

- **Description:** matrice del modello di regressione lineare di dimensione $n \times k$
- **Formula:**

$$X = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{1,k-1} \\ 1 & x_{2,1} & \dots & x_{2,k-1} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,k-1} \end{pmatrix}$$

- **Example:**

```
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> modello <- lm(formula = y ~ x1 + x2 + x3)
> k <- 4
> n <- 8
> X <- model.matrix(object = modello)
> X

  (Intercept)  x1  x2   x3
1           1 1.1 1.2 1.40
2           1 2.3 3.4 5.60
3           1 4.5 5.6 7.56
4           1 6.7 7.5 6.00
5           1 8.9 7.5 5.40
6           1 3.4 6.7 6.60
7           1 5.6 8.6 8.70
8           1 6.7 7.6 8.70
attr(,"assign")
[1] 0 1 2 3
```

kappa()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times m$

  exact = TRUE

- **Description:** calcola il *ConditionNumber* come rapporto tra il maggiore ed il minore valore singolare non nullo della matrice diagonale D
- **Formula:**

$$\frac{\max\left(\text{diag}(D)\right)}{\min\left(\text{diag}(D)\right)}$$

$$\text{dove} \quad A = U\,D\,V^T \quad \text{e} \quad U^T\,U = I_m = V^T\,V = V\,V^T$$

- **Example 1:**

```
> A <- matrix(data = c(1.2, 3, 5.6, 3, 4, 6.7, 5.6, 6.7, 9.8),
+     nrow = 3, ncol = 3)
> 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 = TRUE)

[1] 96.86229
```

- **Example 2:**

```
> A <- matrix(data = c(1, 2, 3, 4, 7, 8, 9, 8), nrow = 4, ncol = 2)
> 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 = TRUE)

[1] 8.923297
```

- **Note:** Calcola il *Condition Number* con la funzione svd().

lower.tri()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times n$

- **Description:** matrice triangolare inferiore di dimensione $n \times n$ a partire dalla matrice A
- **Example 1:**

```
> A <- matrix(data = 1:9, nrow = 3, ncol = 3)
> A

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

> n <- 3
> A[t(lower.tri(A, diag = FALSE))] <- 0
> A

     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    2    5    0
[3,]    3    6    9
```

- **Example 2:**

```
> A <- matrix(data = c(1, 2, 7, 8), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]    1    7
[2,]    2    8

> n <- 2
> A[t(lower.tri(A, diag = FALSE))] <- 0
> A

     [,1] [,2]
[1,]    1    0
[2,]    2    8
```

upper.tri()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times n$

- **Description:** matrice triangolare superiore di dimensione $n \times n$ a partire dalla matrice A
- **Example 1:**

```
> A <- matrix(data = 1:9, nrow = 3, ncol = 3)
> A

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

```
> n <- 3
> A[lower.tri(A, diag = FALSE)] <- 0
> A

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

- **Example 2:**

```
> A <- matrix(data = c(1, 2, 7, 8), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]    1    7
[2,]    2    8

> n <- 2
> A[lower.tri(A, diag = FALSE)] <- 0
> A

     [,1] [,2]
[1,]    1    7
[2,]    0    8
```

backsolve()

- **Package:** base
- **Input:**

  `r` matrice A dei coefficienti di dimensione $n \times n$

  `data` matrice b dei termini noti di dimensione $1 \times n$

  `upper.tri = TRUE / FALSE` sistema triangolare superiore od inferiore

  `transpose = TRUE / FALSE` matrice dei coefficienti trasposta

- **Description:** soluzione di un sistema triangolare di dimensione $n \times n$
- **Formula:**

`upper.tri = TRUE` AND `transpose = TRUE`

$$\left(\begin{array}{ccccc|c} a_{1,1} & 0 & \ldots & \ldots & 0 & b_1 \\ a_{1,2} & a_{2,2} & 0 & \ldots & 0 & b_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ a_{1,n-1} & a_{2,n-1} & \ldots & \ddots & 0 & \vdots \\ a_{1,n} & a_{2,n} & \ldots & \ldots & a_{n,n} & b_n \end{array}\right)$$

`upper.tri = TRUE` AND `transpose = FALSE`

$$\left(\begin{array}{ccccc|c} a_{1,1} & a_{1,2} & \ldots & a_{1,n-1} & a_{1,n} & b_1 \\ 0 & a_{2,2} & \ldots & a_{2,n-1} & a_{2,n} & b_2 \\ \vdots & 0 & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & a_{n,n} & b_n \end{array}\right)$$

`upper.tri = FALSE` AND `transpose = TRUE`

$$\left(\begin{array}{ccccc|c} a_{1,1} & a_{2,1} & \dots & a_{n-1,1} & a_{n,1} & b_1 \\ 0 & a_{2,2} & \dots & a_{n-1,2} & a_{n,2} & b_2 \\ \vdots & 0 & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & a_{n,n} & b_n \end{array}\right)$$

`upper.tri = FALSE` AND `transpose = FALSE`

$$\left(\begin{array}{ccccc|c} a_{1,1} & 0 & \dots & \dots & 0 & b_1 \\ a_{2,1} & a_{2,2} & 0 & \dots & 0 & b_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \dots & \ddots & 0 & \vdots \\ a_{n,1} & a_{n,2} & \dots & \dots & a_{n,n} & b_n \end{array}\right)$$

- **Example 1:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3, byrow = FALSE)
> 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 <- c(8, 4, 2)
> b

[1] 8 4 2

> backsolve(r = A, x = b, upper.tri = TRUE, transpose = TRUE)

[1]  8.000000 -5.000000 -6.016667
```

- **Example 2:**

```
> A <- matrix(data = c(1.2, 0.34, 7.7, 4.5), nrow = 2, ncol = 2,
+     byrow = TRUE)
> A

     [,1] [,2]
[1,]  1.2 0.34
[2,]  7.7 4.50

> b <- c(7.2, -10.4)
> b

[1]   7.2 -10.4

> backsolve(r = A, x = b, upper.tri = FALSE, transpose = FALSE)

[1]   6.00000 -12.57778
```

forwardsolve()

- **Package:** base
- **Input:**

  `l` matrice A dei coefficienti di dimensione $n \times n$

  `x` matrice b dei termini noti di dimensione $1 \times n$

  `upper.tri = TRUE / FALSE` sistema triangolare superiore od inferiore

  `transpose = TRUE / FALSE` matrice dei coefficienti trasposta

- **Description:** soluzione di un sistema triangolare di dimensione $n \times n$
- **Formula:**

`upper.tri = TRUE` AND `transpose = TRUE`

$$\left(\begin{array}{ccccc|c} a_{1,1} & 0 & \dots & \dots & 0 & b_1 \\ a_{1,2} & a_{2,2} & 0 & \dots & 0 & b_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ a_{1,n-1} & a_{2,n-1} & \dots & \ddots & 0 & \vdots \\ a_{1,n} & a_{2,n} & \dots & \dots & a_{n,n} & b_n \end{array}\right)$$

`upper.tri = TRUE` AND `transpose = FALSE`

$$\left(\begin{array}{ccccc|c} a_{1,1} & a_{1,2} & \dots & a_{1,n-1} & a_{1,n} & b_1 \\ 0 & a_{2,2} & \dots & a_{2,n-1} & a_{2,n} & b_2 \\ \vdots & 0 & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & a_{n,n} & b_n \end{array}\right)$$

`upper.tri = FALSE` AND `transpose = TRUE`

$$\left(\begin{array}{ccccc|c} a_{1,1} & a_{2,1} & \dots & a_{n-1,1} & a_{n,1} & b_1 \\ 0 & a_{2,2} & \dots & a_{n-1,2} & a_{n,2} & b_2 \\ \vdots & 0 & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & a_{n,n} & b_n \end{array}\right)$$

`upper.tri = FALSE` AND `transpose = FALSE`

$$\left(\begin{array}{ccccc|c} a_{1,1} & 0 & \dots & \dots & 0 & b_1 \\ a_{2,1} & a_{2,2} & 0 & \dots & 0 & b_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \dots & \ddots & 0 & \vdots \\ a_{n,1} & a_{n,2} & \dots & \dots & a_{n,n} & b_n \end{array}\right)$$

- **Example 1:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3, byrow = FALSE)
> 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 <- c(8, 4, 2)
> b
```

```
[1] 8 4 2
```

```
> forwardsolve(l = A, x = b, upper.tri = TRUE, transpose = TRUE)
```

```
[1]  8.000000 -5.000000 -6.016667
```

- **Example 2:**

```
> A <- matrix(data = c(1.2, 0.34, 7.7, 4.5), nrow = 2, ncol = 2,
+     byrow = TRUE)
> A
```

```
     [,1] [,2]
[1,]  1.2 0.34
[2,]  7.7 4.50
```

```
> b <- c(7.2, -10.4)
> b
```

```
[1]   7.2 -10.4
```

```
> forwardsolve(l = A, x = b, upper.tri = FALSE, transpose = FALSE)
```

```
[1]   6.00000 -12.57778
```

2.4 Fattorizzazioni di Matrici

svd()

- **Package:** base
- **Input:**

  A matrice di dimensione $n \times m$

- **Description:** 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:**

$$A = U\,D\,V^T$$

$$\text{dove} \quad U^T\,U = I_m = V^T\,V = V\,V^T$$

- **Example 1:**

```
> A <- matrix(data = c(11, -2, 3.4, 4.1, 5, 7), nrow = 3, ncol = 2)
> 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
```

- **Example 2:**

```
> A <- matrix(data = c(1, 2, 3.45, 7.8), nrow = 2, ncol = 2)
> 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 1.585646e-18
[2,] 1.585646e-18 1.000000e+00
```

```
> V %*% t(V)
```

```
             [,1]         [,2]
[1,] 1.000000e+00 1.585646e-18
[2,] 1.585646e-18 1.000000e+00
```

```
> U %*% D %*% t(V)
```

```
     [,1] [,2]
[1,]    1 3.45
[2,]    2 7.80
```

qr.Q()

- **Package:** base
- **Input:**

  A matrice di rango pieno di dimensione $n \times m$

- **Description:** matrice Q di dimensione $n \times m$
- **Formula:**

$$A = Q\,R$$

$$\text{dove} \quad Q^T\,Q = I_m$$

- **Example 1:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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
```

- **Example 2:**

```
> A <- matrix(data = c(1, 2, 3.45, 7.8), nrow = 2, ncol = 2)
> 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
- **Input:**

  A matrice di rango pieno di dimensione $n \times m$

- **Description:** matrice R triangolare superiore di dimensione $m \times m$
- **Formula:**

$$A = Q R$$

- **Example 1:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8, 9.9, 1, 12), nrow = 3,
+     ncol = 3)
> 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
```

- **Example 2:**

```
> A <- matrix(data = c(1, 2, 3.45, 7.8), nrow = 2, ncol = 2)
> 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
```

chol()

- **Package:** base
- **Input:**

  A matrice simmetrica definita positiva di dimensione $n \times n$

- **Description:** matrice P triangolare superiore di dimensione $n \times n$
- **Formula:**

$$A = P^T P$$

- **Example 1:**

```
> A <- matrix(data = c(5, 1, 1, 3), nrow = 2, ncol = 2)
> A
```

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

```
> n <- 2
> 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
```

- **Example 2:**

```
> A <- matrix(data = c(1.2, 3.4, 3.4, 11.2), nrow = 2, ncol = 2)
> A
```

```
     [,1] [,2]
[1,]  1.2  3.4
[2,]  3.4 11.2
```

```
> n <- 2
> 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
```

chol2inv()

- **Package:** base
- **Input:**

  `P` matrice P triangolare superiore di dimensione $n \times n$

- **Description:** funzione inversa di `chol()`
- **Formula:**

$$(P^T P)^{-1}$$

- **Example:**

```
> A <- matrix(data = c(5, 1, 1, 3), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]    5    1
[2,]    1    3

> n <- 2
> 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

> 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
```

- **Example 2:**

```
> A <- matrix(data = c(1.2, 3.4, 3.4, 11.2), nrow = 2, ncol = 2)
> A

     [,1] [,2]
[1,]  1.2  3.4
[2,]  3.4 11.2

> n <- 2
> 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

> 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
```

ginv()

- **Package:** MASS
- **Input:**

    A matrice di dimensione $n \times m$

- **Description:** inversa generalizzata A_g di dimensione $m \times n$
- **Formula:**

$$A = A\,A_g\,A$$

- **Example 1:**

```
> A <- matrix(data = c(1, -0.2, 3, 4, 5.6, 7.8), nrow = 3, ncol = 2)
> 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
```

- **Example 2:**

```
> A <- matrix(data = c(1.2, 3.4, 3.4, 11.2), nrow = 2, ncol = 2)
> 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
```

2.5 Creazione di Arrays

array()

- **Package:** base
- **Input:**

  data vettore numerico

  dim dimensione

  dimnames etichette di dimensione

- **Description:** creazione
- **Example:**

```
> etichette <- list(c("A", "B"), c("a", "b"), c("X", "Y"))
> myarray <- array(data = 1:8, dim = c(2, 2, 2), dimnames = etichette)
> myarray
```

```
, , 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

> array(data = 1, dim = c(4, 5))

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

dim()

- **Package:** base
- **Input:**

  x array

- **Description:** dimensione
- **Example:**

```
> 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
- **Input:**

  x array

- **Description:** estrazione di elementi
- **Example:**

```
> 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

> 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 = FALSE]

, , 1

     [,1] [,2]
[1,]    3    5

, , 2

     [,1] [,2]
[1,]    9   11
```

dimnames()

- **Package:** base
- **Input:**

  x array

- **Description:** etichette di dimensione
- **Example:**

```
> 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

Misure ed indici statistici

3.1 Minimo e massimo

min()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n
- **Description:** minimo
- **Formula:**

$$x_{(1)}$$

- **Examples:**

```
> x <- c(4.5, 3.4, 8.7, 3.6)
> min(x)

[1] 3.4

> x <- c(1.1, 3.4, 4.5, 6.4, 4, 3, 4)
> min(x)

[1] 1.1
```

max()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n
- **Description:** massimo
- **Formula:**

$$x_{(n)}$$

- **Examples:**

```
> x <- c(1.2, 2.3, 4.5, 6.5)
> max(x)

[1] 6.5

> x <- c(1.1, 3.4, 4.5, 6.4, 4, 3, 4)
> max(x)

[1] 6.4
```

3.2 Campo di variazione e midrange

range()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

- **Description:** minimo e massimo
- **Formula:**

$$x_{(1)} \qquad x_{(n)}$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8)
> min(x)

[1] 0.8

> max(x)

[1] 3.4

> range(x)

[1] 0.8 3.4

> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> min(x)

[1] 1.2

> max(x)

[1] 6.4

> range(x)

[1] 1.2 6.4
```

range2()

- **Package:** sigma2tools
- **Input:**

  x vettore numerico di dimensione n

- **Description:** campo di variazione
- **Formula:**

$$x_{(n)} - x_{(1)}$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8)
> min(x)

[1] 0.8
```

```
> max(x)
```

```
[1] 3.4
```

```
> max(x) - min(x)
```

```
[1] 2.6
```

```
> range2(x)
```

```
[1] 2.6
```

```
> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> min(x)
```

```
[1] 1.2
```

```
> max(x)
```

```
[1] 6.4
```

```
> max(x) - min(x)
```

```
[1] 5.2
```

```
> range2(x)
```

```
[1] 5.2
```

midrange()

- **Package:** sigma2tools
- **Input:**

  x vettore numerico di dimensione n
- **Description:** midrange
- **Formula:**

$$\left(x_{(1)} + x_{(n)}\right) / 2$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8, 1.77, 7.8)
> min(x)
```

```
[1] 0.8
```

```
> max(x)
```

```
[1] 7.8
```

```
> (min(x) + max(x))/2
```

```
[1] 4.3
```

```
> midrange(x)
```

```
[1] 4.3

> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> min(x)

[1] 1.2

> max(x)

[1] 6.4

> (min(x) + max(x))/2

[1] 3.8

> midrange(x)

[1] 3.8
```

extendrange()

- **Package:** grDevices
- **Input:**

  `x` vettore numerico di dimensione n

  `f` percentuale di estensione α del campo di variazione

- **Description:** campo di variazione
- **Formula:**

$$x_{(1)} - \alpha \left(x_{(n)} - x_{(1)}\right) \qquad x_{(n)} + \alpha \left(x_{(n)} - x_{(1)}\right)$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8)
> alpha <- 0.05
> min(x)

[1] 0.8

> max(x)

[1] 3.4

> min(x) - alpha * (max(x) - min(x))

[1] 0.67

> max(x) + alpha * (max(x) - min(x))

[1] 3.53

> extendrange(x, f = 0.05)

[1] 0.67 3.53
```

```
> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> alpha <- 0.05
> min(x)

[1] 1.2

> max(x)

[1] 6.4

> min(x) - alpha * (max(x) - min(x))

[1] 0.94

> max(x) + alpha * (max(x) - min(x))

[1] 6.66

> extendrange(x, f = 0.05)

[1] 0.94 6.66
```

3.3 Media aritmetica, geometrica ed armonica

mean()

- **Package:** base
- **Input:**

  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

- **Description:** media α-*trimmed*
- **Formula:**

$$\bar{x}_\alpha = \begin{cases} \bar{x} & \text{se } \alpha = 0 \\ \frac{1}{n-2\,\lfloor n\,\alpha \rfloor} \sum_{i=\lfloor n\,\alpha \rfloor+1}^{n-\lfloor n\,\alpha \rfloor} x_{(i)} & \text{se } 0 < \alpha < 0.5 \\ Q_{0.5}(x) & \text{se } \alpha = 0.5 \end{cases}$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8, 10.2, 9.3, 7.34)
> n <- 7
> sum(x)/n

[1] 4.748571

> mean(x, trim = 0)

[1] 4.748571

> x <- c(1, 1.2, 3.4, 0.8, 10.2, 9.3, 7.34)
> x <- sort(x)
> x
```

```
[1]  0.80  1.00  1.20  3.40  7.34  9.30 10.20
```

```
> n <- 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 = 0.26)
```

```
[1] 4.448
```

```
> x <- c(1, 1.2, 3.4, 0.8, 10.2, 9.3, 7.34)
> median(x)
```

```
[1] 3.4
```

```
> mean(x, trim = 0.5)
```

```
[1] 3.4
```

mean.g()

- **Package:** labstatR
- **Input:**

  x vettore numerico di elementi positivi di dimensione n
- **Description:** media geometrica
- **Formula:**

$$\bar{x}_G = \left(\prod_{i=1}^{n} x_i \right)^{1/n} = \exp\left(\frac{1}{n} \sum_{i=1}^{n} \log(x_i) \right)$$

- **Examples:**

```
> x <- c(1.2, 2.3, 4.5, 6.5)
> n <- 4
> prod(x)^(1/n)
```

```
[1] 2.997497
```

```
> mean.g(x)
```

```
[1] 2.997497
```

```
> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> n <- 7
> prod(x)^(1/n)
```

```
[1] 3.434782
```

```
> mean.g(x)
```

```
[1] 3.434782
```

mean.a()

- **Package:** labstatR
- **Input:**

  x vettore numerico di elementi non nulli di dimensione n

- **Description:** media armonica
- **Formula:**

$$\bar{x}_A = \left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{x_i} \right)^{-1}$$

- **Examples:**

```
> x <- c(1.2, 2.3, 4.5, 6.5)
> 1/mean(1/x)

[1] 2.432817

> mean.a(x)

[1] 2.432817

> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> 1/mean(1/x)

[1] 2.992404

> mean.a(x)

[1] 2.992404
```

3.4 Mediana e quantili

median()

- **Package:** stats
- **Input:**

  x vettore numerico di dimensione n

- **Description:** mediana
- **Formula:**

$$Q_{0.5}(x) = \begin{cases} x_{(\frac{n+1}{2})} & \text{se } n \text{ è dispari} \\ 0.5 \left(x_{(\frac{n}{2})} + x_{(\frac{n}{2}+1)} \right) & \text{se } n \text{ è pari} \end{cases}$$

- **Examples:**

```
> x <- c(1.2, 0.34, 5.6, 7.4, 2.1, 3.2, 9.87, 10.1)
> x <- sort(x)
> x

[1]  0.34  1.20  2.10  3.20  5.60  7.40  9.87 10.10

> n <- 8
> 0.5 * (x[n/2] + x[n/2 + 1])
```

```
[1] 4.4

> median(x)

[1] 4.4

> x <- c(1.2, 0.34, 5.6, 7.4, 2.1, 3.2, 9.87)
> x <- sort(x)
> x

[1] 0.34 1.20 2.10 3.20 5.60 7.40 9.87

> n <- 7
> x[(n + 1)/2]

[1] 3.2

> median(x)

[1] 3.2
```

- **Note:** Equivale alla funzione `quantile()` quando questa è calcolata in `probs = 0.5`.

quantile()

- **Package:** `stats`
- **Input:**

  `x` vettore numerico di dimensione n

  `probs` valore p di probabilità
- **Description:** quantile al $(100\,p)$%
- **Formula:**

$$Q_p(x) = \begin{cases} x_{(\alpha)} & \text{se } \alpha \text{ è intero} \\ x_{(\lfloor\alpha\rfloor)} + (\alpha - \lfloor\alpha\rfloor)\left(x_{(\lfloor\alpha\rfloor+1)} - x_{(\lfloor\alpha\rfloor)}\right) & \text{se } \alpha \text{ non è intero} \end{cases}$$

$$\text{dove} \quad \alpha = 1 + (n-1)\,p$$

- **Examples:**

```
> x <- c(1.2, 2.3, 0.11, 4.5, 2.3, 4.55, 7.8, 6.6, 9.9)
> x <- sort(x)
> x

[1] 0.11 1.20 2.30 2.30 4.50 4.55 6.60 7.80 9.90

> n <- 9
> p <- 0.25
> alpha <- 1 + (n - 1) * p
> alpha

[1] 3

> x[alpha]

[1] 2.3
```

```
> quantile(x, probs = 0.25)

25%
2.3

> x <- c(1.2, 2.3, 0.11, 4.5)
> x <- sort(x)
> x

[1] 0.11 1.20 2.30 4.50

> n <- 4
> p <- 0.34
> alpha <- 1 + (n - 1) * p
> alpha

[1] 2.02

> x[floor(alpha)] + (alpha - floor(alpha)) * (x[floor(alpha) +
+     1] - x[floor(alpha)])

[1] 1.222

> quantile(x, probs = 0.34)

  34%
1.222

> x <- c(1.2, 4.2, 4.5, -5.6, 6.5, 1.2)
> x <- sort(x)
> n <- 6
> p <- 0.68
> alpha <- 1 + (n - 1) * p
> alpha

[1] 4.4

> x[floor(alpha)] + (alpha - floor(alpha)) * (x[floor(alpha) +
+     1] - x[floor(alpha)])

[1] 4.32

> quantile(x, probs = 0.68)

 68%
4.32
```

- **Note 1:** Equivale alla funzione `median()` quando `probs = 0.5`.
- **Note 2:** Equivale alla funzione `min()` quando `probs = 0`.
- **Note 3:** Equivale alla funzione `max()` quando `probs = 1`.

3.5 Differenza interquartile e deviazione assoluta dalla mediana

IQR()

- **Package:** stats
- **Input:**

  x vettore numerico di dimensione n

- **Description:** differenza interquartile
- **Formula:**

$$IQR(x) = Q_{0.75}(x) - Q_{0.25}(x)$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8, 10.2, 9.3, 7.34)
> diff(quantile(x, probs = c(0.25, 0.75)))

 75%
7.22

> IQR(x)

[1] 7.22

> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> diff(quantile(x, probs = c(0.25, 0.75)))

 75%
1.05

> IQR(x)

[1] 1.05
```

- **Note:** Calcola i quartili con la funzione quantile().

mad()

- **Package:** stats
- **Input:**

  x vettore numerico di dimensione n

  center parametro rispetto al quale si effettuano gli scarti

  constant il valore α della costante positiva

- **Description:** deviazione assoluta dalla mediana
- **Formula:**

$$\alpha \, Q_{0.5} \left(\, | \, x - \text{center}(x) \, | \, \right)$$

- **Examples:**

```
> x <- c(1.2, 3.4, 4.5, 6.4, 4)
> alpha <- 1.23
> alpha * median(abs(x - median(x)))

[1] 0.738

> mad(x, center = median(x), constant = 1.23)
```

```
[1] 0.738

> x <- c(1, 1.2, 3.4, 0.8, 10.2, 9.3, 7.34)
> alpha <- 1.55
> alpha * median(abs(x - mean(x)))

[1] 5.810286

> mad(x, center = mean(x), constant = 1.55)

[1] 5.810286

> x <- c(1.2, 4.2, 4.5, -5.6, 6.5, 1.2)
> alpha <- 2.42
> alpha * median(abs(x - mean(x)))

[1] 5.687

> mad(x, center = mean(x), constant = 2.42)

[1] 5.687
```

- **Note:** Per default vale `constant = 1.4826` = $1 \,/\, \Phi^{-1}(0.75)$ e `center = median(x)`.

3.6 Asimmetria e curtosi

skew()

- **Package:** labstatR
- **Input:**

  x vettore numerico di dimensione n

- **Description:** asimmetria nella popolazione
- **Formula:**

$$\gamma_3 = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{\sigma_x} \right)^3$$

- **Examples:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> sigmax <- sqrt(mean((x - mean(x))^2))
> mean((x - mean(x))^3/sigmax^3)

[1] 0.1701538

> skew(x)

[1] 0.1701538

> x <- c(1.2, 3.4, 5.2, 3.4, 4.4)
> sigmax <- sqrt(mean((x - mean(x))^2))
> mean((x - mean(x))^3/sigmax^3)

[1] -0.5845336

> skew(x)

[1] -0.5845336
```

skewness()

- **Package:** fBasics
- **Input:**

  x vettore numerico di dimensione n

- **Description:** asimmetria campionaria
- **Formula:**

$$\hat{\gamma}_3 = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x} \right)^3$$

- **Examples:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> mean((x - mean(x))^3/sd(x)^3)

[1] 0.1217521

> skewness(x)

[1] 0.1217521
attr(,"method")
[1] "moment"

> x <- c(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
attr(,"method")
[1] "moment"
```

skewness()

- **Package:** e1071
- **Input:**

  x vettore numerico di dimensione n

- **Description:** asimmetria campionaria
- **Formula:**

$$\hat{\gamma}_3 = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x} \right)^3$$

- **Examples:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> mean((x - mean(x))^3/sd(x)^3)

[1] 0.1217521

> skewness(x)
```

```
[1] 0.1217521
attr(,"method")
[1] "moment"
```

```
> x <- c(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
attr(,"method")
[1] "moment"
```

kurt()

- **Package:** labstatR
- **Input:**

  x vettore numerico di dimensione n

- **Description:** kurtosi nella popolazione
- **Formula:**

$$\gamma_4 = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{\sigma_x} \right)^4$$

- **Examples:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> sigmax <- sqrt(mean((x - mean(x))^2))
> mean((x - mean(x))^4/sigmax^4)
```

```
[1] 1.623612
```

```
> kurt(x)
```

```
[1] 1.623612
```

```
> x <- c(1.2, 3.4, 5.2, 3.4, 4.4)
> sigmax <- sqrt(mean((x - mean(x))^2))
> mean((x - mean(x))^4/sigmax^4)
```

```
[1] 2.312941
```

```
> kurt(x)
```

```
[1] 2.312941
```

kurtosis()

- **Package:** fBasics
- **Input:**

  x vettore numerico di dimensione n

- **Description:** kurtosi campionaria
- **Formula:**

$$\hat{\gamma}_4 = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x} \right)^4 - 3$$

- **Examples:**

```
> x <- c(1, 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
attr(,"method")
[1] "excess"

> x <- c(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
attr(,"method")
[1] "excess"
```

kurtosis()

- **Package:** e1071
- **Input:**

  x vettore numerico di dimensione n

- **Description:** kurtosi campionaria
- **Formula:**

$$\hat{\gamma}_4 = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x} \right)^4 - 3$$

- **Examples:**

```
> x <- c(1, 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
attr(,"method")
[1] "excess"
```

```
> x <- c(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
attr(,"method")
[1] "excess"
```

geary()

- **Package:**
- **Input:**

  x vettore numerico di dimensione n

- **Description:** kurtosi secondo *Geary*
- **Formula:**

$$\gamma_4^G = \frac{1}{n}\sum_{i=1}^{n}\frac{|\,x_i - \bar{x}\,|}{\sigma_x}$$

- **Examples:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> sigmax <- sqrt(mean((x - mean(x))^2))
> mean(abs(x - mean(x))/sigmax)
```

```
[1] 0.8702836
```

```
> geary(x)
```

```
[1] 0.8702836
```

```
> x <- c(1.2, 3.4, 5.2, 3.4, 4.4)
> sigmax <- sqrt(mean((x - mean(x))^2))
> mean(abs(x - mean(x))/sigmax)
```

```
[1] 0.7629055
```

```
> geary(x)
```

```
[1] 0.7629055
```

3.7 Coefficiente di variazione

var.coeff()

- **Package:** ineq
- **Input:**

  x vettore numerico di dimensione n

  square = TRUE / FALSE quadrato

- **Description:** coefficiente di variazione nella popolazione
- **Formula:**

square = FALSE

$$CV_x = \sigma_x \,/\, \bar{x}$$

square = TRUE

$$CV_x^2 = (\sigma_x \,/\, \bar{x})^2$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8)
> sigmax <- sqrt(mean((x - mean(x))^2))
> sigmax/mean(x)

[1] 0.6555055

> var.coeff(x, square = FALSE)

[1] 0.6555055

> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> sigmax <- sqrt(mean((x - mean(x))^2))
> (sigmax/mean(x))^2

[1] 0.1484087

> var.coeff(x, square = TRUE)

[1] 0.1484087
```

cv()

- **Package:** labstatR
- **Input:**

  x vettore numerico di dimensione n

- **Description:** coefficiente di variazione nella popolazione
- **Formula:**

$$CV_x = \sigma_x \,/\, |\,\bar{x}\,| = \sqrt{\frac{n-1}{n}}\, cv_x$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8)
> sigmax <- sqrt(mean((x - mean(x))^2))
> sigmax/abs(mean(x))

[1] 0.6555055

> cv(x)

[1] 0.6555055

> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> sigmax <- sqrt(mean((x - mean(x))^2))
> sigmax/abs(mean(x))

[1] 0.3852385

> cv(x)

[1] 0.3852385
```

cv2()

- **Package:** sigma2tools
- **Input:**

  x vettore numerico di dimensione n

- **Description:** coefficiente di variazione campionario
- **Formula:**

$$cv_x = s_x \,/\, |\,\bar{x}\,| = \sqrt{\frac{n}{n-1}}\, CV_x$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8)
> sd(x)/abs(mean(x))

[1] 0.7569126

> cv2(x)

[1] 0.7569126

> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> sd(x)/abs(mean(x))

[1] 0.4161051

> cv2(x)

[1] 0.4161051
```

3.8 Scarto quadratico medio e deviazione standard

sigma()

- **Package:** sigma2tools
- **Input:**

  x vettore numerico di dimensione n

- **Description:** scarto quadratico medio
- **Formula:**

$$\sigma_x = \left(\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right)^{1\,/\,2} = \sqrt{\frac{1}{n}\, ss_x} = \sqrt{\frac{n-1}{n}}\, s_x$$

- **Examples:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> sqrt(mean((x - mean(x))^2))

[1] 2.868031

> sigma(x)

[1] 2.868031

> x <- c(1.2, 2.3, 4.5, 6.5)
> sqrt(mean((x - mean(x))^2))

[1] 2.041292

> sigma(x)

[1] 2.041292
```

sd()

- **Package:** stats
- **Input:**

  x vettore numerico di dimensione n

- **Description:** deviazione standard
- **Formula:**

$$s_x = \left(\frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right)^{1\,/\,2} = \sqrt{\frac{1}{n-1}\, ss_x} = \sqrt{\frac{n}{n-1}}\, \sigma_x$$

- **Examples:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> n <- 5
> sqrt(sum((x - mean(x))^2)/(n - 1))

[1] 3.206556

> sd(x)

[1] 3.206556
```

```
> x <- c(1.3, 4.2, 3.3, 8.7)
> n <- 4
> sqrt(sum((x - mean(x))^2)/(n - 1))
```

```
[1] 3.127699
```

```
> sd(x)
```

```
[1] 3.127699
```

3.9 Errore standard

popstderror()

- **Package:** sigma2tools
- **Input:**

  x vettore numerico di dimensione n

- **Description:** errore standard nella popolazione
- **Formula:**

$$SE_x = \sigma_x \,/\, \sqrt{n} = \sqrt{\frac{n-1}{n}}\, se_x$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8)
> n <- 4
> sigmax <- sqrt(sum((x - mean(x))^2)/n)
> sigmax/sqrt(n)
```

```
[1] 0.5244044
```

```
> popstderror(x)
```

```
[1] 0.5244044
```

```
> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> n <- 7
> sigmax <- sqrt(sum((x - mean(x))^2)/n)
> sigmax/sqrt(n)
```

```
[1] 0.5512245
```

```
> popstderror(x)
```

```
[1] 0.5512245
```

stderror()

- **Package:** sigma2tools
- **Input:**

  x vettore numerico di dimensione n

- **Description:** errore standard campionario
- **Formula:**

$$se_x = s_x \,/\, \sqrt{n} = \sqrt{\frac{n}{n-1}}\, SE_x$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8)
> n <- 4
> sd(x)/sqrt(n)

[1] 0.6055301

> stderror(x)

[1] 0.6055301

> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> n <- 7
> sd(x)/sqrt(n)

[1] 0.5953905

> stderror(x)

[1] 0.5953905
```

3.10 Varianza e devianza

sigma2()

- **Package:** labstatR
- **Input:**

  x vettore numerico di dimensione n

- **Description:** varianza nella popolazione
- **Formula:**

$$\sigma_x^2 = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 = \frac{1}{n} \sum_{i=1}^{n} x_i^2 - \bar{x}^2 = \frac{1}{n}\, ss_x = \frac{n-1}{n}\, s_x^2$$

- **Examples:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> mean((x - mean(x))^2)

[1] 8.2256

> sigma2(x)

[1] 8.2256
```

```
> x <- c(1.2, 2.3, 4.5, 6.5)
> mean((x - mean(x))^2)

[1] 4.166875

> sigma2(x)

[1] 4.166875
```

var()

- **Package:** fUtilities
- **Input:**

  x vettore numerico di dimensione n

- **Description:** varianza campionaria
- **Formula:**

$$s_x^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 = \frac{1}{n-1} \sum_{i=1}^{n} x_i^2 - \frac{n}{n-1} \bar{x}^2 = \frac{1}{n-1} ss_x = \frac{n}{n-1} \sigma_x^2$$

- **Examples:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> n <- 5
> sum((x - mean(x))^2)/(n - 1)

[1] 10.282

> var(x)

[1] 10.282

> x <- c(1.2, 3.4, 5.6, 3.7, 7.8, 8.5)
> n <- 6
> sum((x - mean(x))^2)/(n - 1)

[1] 7.826667

> var(x)

[1] 7.826667
```

ssdev()

- **Package:** sigma2tools
- **Input:**

  x vettore numerico di dimensione n

- **Description:** devianza
- **Formula:**

$$ss_x = \sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - n\,\bar{x}^2 = (n-1)\,s_x^2 = n\,\sigma_x^2$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8)
> sum((x - mean(x))^2)

[1] 4.4

> ssdev(x)

[1] 4.4

> x <- c(1.2, 2.3, 4.5, 6.5)
> sum((x - mean(x))^2)

[1] 16.6675

> ssdev(x)

[1] 16.6675
```

3.11 Covarianza e codevianza

COV()

- **Package:** labstatR
- **Input:**

  x vettore numerico di dimensione n

  y vettore numerico di dimensione n

- **Description:** covarianza nella popolazione
- **Formula:**

$$\sigma_{xy} = \frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y}) = \frac{1}{n}\sum_{i=1}^{n} x_i\,y_i - \bar{x}\,\bar{y} = \frac{1}{n}\,ss_{xy} = \frac{n-1}{n}\,s_{xy}$$

- **Examples:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> y <- c(1.2, 3.4, 4.5, 6.4, 4)
> mean((x - mean(x)) * (y - mean(y)))

[1] 3.298

> COV(x, y)

[1] 3.298
```

```
> x <- c(1.2, 3.4, 5.6, 7.5, 7.7, 7.8)
> y <- c(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:** fUtilities
- **Input:**

  x vettore numerico di dimensione n

  y vettore numerico di dimensione n

- **Description:** covarianza campionaria
- **Formula:**

$$s_{xy} = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y}) = \frac{1}{n-1} \sum_{i=1}^{n} x_i\, y_i - \frac{n}{n-1}\, \bar{x}\, \bar{y} = \frac{1}{n-1}\, ss_{xy} = \frac{n}{n-1}\, \sigma_{xy}$$

- **Examples:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> y <- c(1.3, 4.2, 3.3, 8.7, 3.7)
> n <- 5
> sum((x - mean(x)) * (y - mean(y)))/(n - 1)

[1] 4.4535

> cov(x, y)

[1] 4.4535

> x <- c(1.5, 6.4, 6.3, 6.7, 7.5, 4.5, 4.2, 7.8)
> y <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4, 3.4)
> n <- 8
> sum((x - mean(x)) * (y - mean(y)))/(n - 1)

[1] 1.970893

> cov(x, y)

[1] 1.970893
```

codev()

- **Package:** sigma2tools
- **Input:**

  x vettore numerico di dimensione n

  y vettore numerico di dimensione n
- **Description:** codevianza
- **Formula:**

$$ss_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y}) = \sum_{i=1}^{n} x_i\, y_i - n\, \bar{x}\, \bar{y} = (n-1)\, s_{xy} = n\, \sigma_{xy}$$

- **Examples:**

```
> x <- c(1.5, 6.4, 6.3, 6.7, 7.5)
> y <- c(1.2, 3.4, 4.5, 6.4, 4)
> sum((x - mean(x)) * (y - mean(y)))

[1] 14.03

> codev(x, y)

[1] 14.03

> x <- c(1.2, 3.4, 5.6, 7.5, 7.7, 7.8)
> y <- c(1.1, 2.3, 4.4, 5.1, 2.9, 8.7)
> sum((x - mean(x)) * (y - mean(y)))

[1] 26.65333

> codev(x, y)

[1] 26.65333
```

3.12 Matrice di varianza e covarianza

sigma2m()

- **Package:** sigma2tools
- **Input:**

  x matrice di dimensione $n \times k$ le cui colonne corrispondono ai vettori numerici $x_1, x_2, \ldots, x_k$
- **Description:** matrice di covarianza non corretta
- **Formula:**

$$s_{x_i x_j} = \frac{1}{n}\, (x_i - \bar{x}_i)^T\, (x_j - \bar{x}_j) \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 2
> x1 <- c(1.3, 4.6, 7.7, 8.4, 12.4)
> x2 <- c(1.2, 3.4, 4.5, 6.4, 4)
> n <- 5
> (n - 1) * var(x1)/n

[1] 13.9576
```

```
> (n - 1) * var(x2)/n

[1] 2.832

> (n - 1) * cov(x1, x2)/n

[1] 4.21

> x <- cbind(x1, x2)
> sigma2m(x)

         x1    x2
x1 13.9576 4.210
x2  4.2100 2.832

> k <- 3
> x1 <- c(1.1, 3.6, 7.4, 6.8, 9.8, 7.6, 3.8)
> x2 <- c(5.6, 7.54, 7.3, 3.5, 6.45, 5.4, 3.4)
> x3 <- c(2.8, 8.5, 6.4, 7.8, 98.6, 7.5, 5.7)
> n <- 7
> (n - 1) * var(x1)/n

[1] 7.670612

> (n - 1) * var(x2)/n

[1] 2.380869

> (n - 1) * var(x3)/n

[1] 1042.793

> (n - 1) * cov(x1, x2)/n

[1] 0.5416122

> (n - 1) * cov(x1, x3)/n

[1] 56.06959

> (n - 1) * cov(x2, x3)/n

[1] 11.56516

> x <- cbind(x1, x2, x3)
> sigma2m(x)

           x1         x2         x3
x1  7.6706122  0.5416122   56.06959
x2  0.5416122  2.3808694   11.56516
x3 56.0695918 11.5651633 1042.79265
```

- **Note:** Naturalmente vale che $s_{x_i x_i} = s^2_{x_i} \quad \forall\, i = 1, 2, \ldots, k$.

Var()

- **Package:** car
- **Input:**

  x matrice di dimensione $n \times k$ le cui colonne corrispondono ai vettori numerici $x_1, x_2, \ldots, x_k$

  diag = TRUE / FALSE varianze campionarie o matrice di covarianza

- **Description:** matrice di covarianza
- **Formula:**

diag = TRUE

$$s_{x_i}^2 = \frac{1}{n-1} (x_i - \bar{x}_i)^T (x_i - \bar{x}_i) \quad \forall\, i = 1, 2, \ldots, k$$

diag = FALSE

$$s_{x_i x_j} = \frac{1}{n-1} (x_i - \bar{x}_i)^T (x_j - \bar{x}_j) \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 2
> x1 <- c(0.5, -0.1, 0.2, -1.9, 1.9, 0.7, -1.5, 0, -2.5, 1.6, 0.2,
+     -0.3)
> x2 <- c(1, 4, 10, 2.1, 3.5, 5.6, 8.4, 12, 6.5, 2, 1.2, 3.4)
> n <- 12
> var(x1)

[1] 1.734545

> var(x2)

[1] 12.89295

> cov(x1, x2)

[1] -1.070909

> x <- cbind(x1, x2)
> Var(x, diag = TRUE)

       x1        x2
 1.734545 12.892955

> Var(x, diag = FALSE)

          x1        x2
x1  1.734545 -1.070909
x2 -1.070909 12.892955

> k <- 3
> x1 <- c(1.2, 3.4, 5.6, 7.5, 7.7)
> x2 <- c(1.1, 2.1, 4.2, 5.3, 3.3)
> x3 <- c(1, 2.6, 7.6, 7.7, 7.7)
> n <- 5
> var(x1)

[1] 7.717
```

```
> var(x2)

[1] 2.76

> var(x3)

[1] 10.647

> cov(x1, x2)

[1] 3.965

> cov(x1, x3)

[1] 8.628

> cov(x2, x3)

[1] 4.895

> x <- cbind(x1, x2, x3)
> Var(x, diag = TRUE)

    x1     x2     x3
 7.717  2.760 10.647

> Var(x, diag = FALSE)

      x1    x2     x3
x1 7.717 3.965  8.628
x2 3.965 2.760  4.895
x3 8.628 4.895 10.647
```

- **Note:** Naturalmente vale che $s_{x_i x_i} = s^2_{x_i} \quad \forall\, i = 1, 2, \ldots, k$.

3.13 Correlazione di Pearson, Spearman e Kendall

cor()

- **Package:** `fUtilities`
- **Input:**

  `x` vettore numerico di dimensione n

  `y` vettore numerico di dimensione n

  `method = "pearson" / "spearman" / "kendall"` tipo di coefficiente
- **Description:** coefficiente di correlazione
- **Formula:**

`method = "pearson"`

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\,(y_i - \bar{y})}{\left(\sum_{i=1}^{n}(x_i - \bar{x})^2\right)^{1/2}\left(\sum_{i=1}^{n}(y_i - \bar{y})^2\right)^{1/2}} = \frac{\sum_{i=1}^{n} x_i\, y_i - n\,\bar{x}\,\bar{y}}{\left(\sum_{i=1}^{n} x_i^2 - n\,\bar{x}^2\right)^{1/2}\left(\sum_{i=1}^{n} y_i^2 - n\,\bar{y}^2\right)^{1/2}}$$

`method = "spearman"`

$$r_{xy}^S = \frac{\sum_{i=1}^n (a_i - \bar{a})\,(b_i - \bar{b})}{\left(\sum_{i=1}^n (a_i - \bar{a})^2\right)^{1\,/\,2} \left(\sum_{i=1}^n (b_i - \bar{b})^2\right)^{1\,/\,2}} = \frac{\sum_{i=1}^n a_i\, b_i - n\,((n+1)\,/\,2)^2}{\left(\sum_{i=1}^n a_i^2 - n\,((n+1)\,/\,2)^2\right)^{1\,/\,2} \left(\sum_{i=1}^n b_i^2 - n\,((n+1)\,/\,2)^2\right)^{1\,/\,2}}$$

dove a, b sono i ranghi di x ed y rispettivamente.

`method = "kendall"`

$$r_{xy}^K = \frac{2\,\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \text{sign}((x_j - x_i)\,(y_j - y_i))}{\left(n\,(n-1) - \sum_{i=1}^{g} t_i\,(t_i - 1)\right)^{1\,/\,2} \left(n\,(n-1) - \sum_{j=1}^{h} u_j\,(u_j - 1)\right)^{1\,/\,2}}$$

dove t, u sono i ties di x ed y rispettivamente.

- **Examples:**

```
> x <- c(1, 2, 2, 4, 3, 3)
> y <- c(6, 6, 7, 7, 7, 9)
> cov(x, y)/(sd(x) * sd(y))

[1] 0.522233

> cor(x, y, method = "pearson")

[1] 0.522233

> x <- c(1, 2, 3, 5.6, 7.6, 2.3, 1)
> y <- c(1.2, 2.2, 3, 15.6, 71.6, 2.2, 1.2)
> a <- rank(x)
> b <- rank(y)
> rhoS <- cov(a, b)/(sd(a) * sd(b))
> rhoS

[1] 0.9908674

> cor(x, y, method = "spearman")

[1] 0.9908674

> x <- c(1, 2, 2, 4, 3, 3)
> y <- c(6, 6, 7, 7, 7, 9)
> n <- 6
> matrice <- matrix(0, nrow = n - 1, ncol = n, byrow = FALSE)
> for (i in 1:(n - 1)) for (j in (i + 1):n) matrice[i, j] <- sign((x[j] -
+     x[i]) * (y[j] - y[i]))
> 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

> rhoK <- (2 * sum(matrice))/((n * (n - 1) - sum(t * (t - 1)))^0.5 *
+     (n * (n - 1) - sum(u * (u - 1)))^0.5)
> rhoK

[1] 0.5853694

> cor(x, y, method = "kendall")

[1] 0.5853694

> x <- c(1, 2, 3, 5.6, 7.6, 2.3, 1)
> y <- c(1.2, 2.2, 3, 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

> x <- c(1, 2, 2, 4, 3, 3)
> y <- c(6, 6, 7, 7, 7, 9)
> a <- rank(x)
> b <- rank(y)
> rhoS <- cov(a, b)/(sd(a) * sd(b))
> rhoS

[1] 0.6833149

> cor(x, y, method = "spearman")

[1] 0.6833149

> x <- c(1, 2, 3, 5.6, 7.6, 2.3, 1)
> y <- c(1.2, 2.2, 3, 15.6, 71.6, 2.2, 1.2)
> n <- 7
> matrice <- matrix(0, nrow = n - 1, ncol = n, byrow = FALSE)
> for (i in 1:(n - 1)) for (j in (i + 1):n) matrice[i, j] <- sign((x[j] -
+     x[i]) * (y[j] - y[i]))
> 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

> rhoK <- (2 * sum(matrice))/((n * (n - 1) - sum(t * (t - 1)))^0.5 *
+     (n * (n - 1) - sum(u * (u - 1)))^0.5)
> rhoK

[1] 0.9746794

> cor(x, y, method = "kendall")

[1] 0.9746794
```

cov2cor()

- **Package:** stats
- **Input:**

  V matrice di covarianza di dimensione $k \times k$ relativa ai vettori numerici $x_1, x_2, \ldots, x_k$
- **Description:** converte la matrice di covarianza nella matrice di correlazione
- **Formula:**

$$r_{x_i x_j} = \frac{\sigma_{x_i x_j}}{\sigma_{x_i} \, \sigma_{x_j}} = \frac{s_{x_i x_j}}{s_{x_i} \, s_{x_j}} = \frac{ss_{x_i x_j}}{\sqrt{ss_{x_i} \, ss_{x_j}}} \quad \forall \, i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x1 <- c(-1.2, -1.3, -6.7, 0.8, -7.6, -5.6)
> x2 <- c(1, 2, 3, 5, 6, 7.3)
> dati <- cbind(x1, x2)
> dati

       x1  x2
[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

> n <- 6
> k <- 2
> V <- cov(dati)
> V

       x1     x2
x1 12.004 -3.780
x2 -3.780  5.975
```

```
> cor(dati)

           x1         x2
x1  1.0000000 -0.4463339
x2 -0.4463339  1.0000000

> cov2cor(V)

           x1         x2
x1  1.0000000 -0.4463339
x2 -0.4463339  1.0000000

> x1 <- c(1, 2, 4.5, 1.2, 1.23)
> x2 <- c(2.7, -7.8, 8.8, 4.5, 5.21)
> x3 <- c(1, 4.77, 8.9, 7.8, 0.8)
> dati <- cbind(x1, x2, x3)
> dati

       x1    x2   x3
[1,] 1.00  2.70 1.00
[2,] 2.00 -7.80 4.77
[3,] 4.50  8.80 8.90
[4,] 1.20  4.50 7.80
[5,] 1.23  5.21 0.80

> n <- 5
> k <- 3
> V <- cov(dati)
> V

         x1        x2        x3
x1 2.120480  2.969010  3.679945
x2 2.969010 39.249620  5.167965
x3 3.679945  5.167965 14.036080

> cor(dati)

          x1        x2        x3
x1 1.0000000 0.3254444 0.6745301
x2 0.3254444 1.0000000 0.2201805
x3 0.6745301 0.2201805 1.0000000

> cov2cor(V)

          x1        x2        x3
x1 1.0000000 0.3254444 0.6745301
x2 0.3254444 1.0000000 0.2201805
x3 0.6745301 0.2201805 1.0000000
```

- **Note:** Naturalmente vale che $s_{x_i x_i} = s^2_{x_i} \quad \forall\, i = 1, 2, \ldots, k$.

cancor()

- **Package:** stats
- **Input:**

  `x` vettore numerico di dimensione n

  `y` vettore numerico di dimensione n

  `xcenter = TRUE / FALSE` parametro di posizione

  `ycenter = TRUE / FALSE` parametro di posizione

- **Description:** correlazione canonica
- **Output:**

  `cor` coefficiente di correlazione

  `xcenter` parametro di locazione

  `ycenter` parametro di locazione

- **Formula:**

  `cor`

  `xcenter = TRUE AND ycenter = TRUE`

$$r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y})}{\left(\sum_{i=1}^{n} (x_i - \bar{x})^2\right)^{1/2} \left(\sum_{i=1}^{n} (y_i - \bar{y})^2\right)^{1/2}}$$

  `xcenter = TRUE AND ycenter = FALSE`

$$r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})\,y_i}{\left(\sum_{i=1}^{n} (x_i - \bar{x})^2\right)^{1/2} \left(\sum_{i=1}^{n} y_i^2\right)^{1/2}}$$

  `xcenter = FALSE AND ycenter = TRUE`

$$r_{xy} = \frac{\sum_{i=1}^{n} x_i\,(y_i - \bar{y})}{\left(\sum_{i=1}^{n} x_i^2\right)^{1/2} \left(\sum_{i=1}^{n} (y_i - \bar{y})^2\right)^{1/2}}$$

  `xcenter = FALSE AND ycenter = FALSE`

$$r_{xy} = \frac{\sum_{i=1}^{n} x_i\,y_i}{\left(\sum_{i=1}^{n} x_i^2\right)^{1/2} \left(\sum_{i=1}^{n} y_i^2\right)^{1/2}}$$

  `xcenter`

  `xcenter = TRUE`

$$\bar{x}$$

  `xcenter = FALSE`

$$0$$

  `ycenter`

  `ycenter = TRUE`

$$\bar{y}$$

  `ycenter = FALSE`

$$0$$

- **Examples:**

```
> x <- c(1, 2, 3, 5.6, 7.6, 2.3, 1)
> y <- c(1.2, 2.2, 3, 15.6, 71.6, 2.2, 1.2)
> n <- 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 = TRUE, ycenter = TRUE)$cor

[1] 0.8790885

> mean(x)

[1] 3.214286

> cancor(x, y, xcenter = TRUE, ycenter = TRUE)$xcenter

[1] 3.214286

> mean(y)

[1] 13.85714

> cancor(x, y, xcenter = TRUE, ycenter = TRUE)$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 = TRUE, ycenter = FALSE)$cor

[1] 0.7616638

> mean(x)

[1] 3.214286

> cancor(x, y, xcenter = TRUE, ycenter = FALSE)$xcenter

[1] 3.214286

> cancor(x, y, xcenter = TRUE, ycenter = FALSE)$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 = FALSE, ycenter = TRUE)$cor

[1] 0.5118281

> cancor(x, y, xcenter = FALSE, ycenter = TRUE)$xcenter

[1] 0

> mean(y)

[1] 13.85714
```

```
> cancor(x, y, xcenter = FALSE, ycenter = TRUE)$ycenter

[1] 13.85714

> sum(x * y)/(sum(x^2)^0.5 * sum(y^2)^0.5)

[1] 0.8494115

> cancor(x, y, xcenter = FALSE, ycenter = FALSE)$cor

[1] 0.8494115

> cancor(x, y, xcenter = FALSE, ycenter = FALSE)$xcenter

[1] 0

> cancor(x, y, xcenter = FALSE, ycenter = FALSE)$ycenter

[1] 0

> x <- c(1.2, 2.3, 4.5, 3.2, 4.7)
> y <- c(1.8, 9.87, 7.5, 6.6, 7.7)
> n <- 5
> sum((x - mean(x)) * (y - mean(y)))/(sum((x - mean(x))^2)^0.5 *
+     sum((y - mean(y))^2)^0.5)

[1] 0.536735

> cancor(x, y, xcenter = TRUE, ycenter = TRUE)$cor

[1] 0.536735

> mean(x)

[1] 3.18

> cancor(x, y, xcenter = TRUE, ycenter = TRUE)$xcenter

[1] 3.18

> mean(y)

[1] 6.694

> cancor(x, y, xcenter = TRUE, ycenter = TRUE)$ycenter

[1] 6.694

> sum((x - mean(x)) * y)/(sum((x - mean(x))^2)^0.5 * sum(y^2)^0.5)

[1] 0.1990048

> cancor(x, y, xcenter = TRUE, ycenter = FALSE)$cor

[1] 0.1990048

> mean(x)
```

```
[1] 3.18

> cancor(x, y, xcenter = TRUE, ycenter = FALSE)$xcenter

[1] 3.18

> cancor(x, y, xcenter = TRUE, ycenter = FALSE)$ycenter

[1] 0

> sum(x * (y - mean(y)))/(sum(x^2)^0.5 * sum((y - mean(y))^2)^0.5)

[1] 0.2061343

> cancor(x, y, xcenter = FALSE, ycenter = TRUE)$cor

[1] 0.2061343

> cancor(x, y, xcenter = FALSE, ycenter = TRUE)$xcenter

[1] 0

> mean(y)

[1] 6.694

> cancor(x, y, xcenter = FALSE, ycenter = TRUE)$ycenter

[1] 6.694

> sum(x * y)/(sum(x^2)^0.5 * sum(y^2)^0.5)

[1] 0.9339306

> cancor(x, y, xcenter = FALSE, ycenter = FALSE)$cor

[1] 0.9339306

> cancor(x, y, xcenter = FALSE, ycenter = FALSE)$xcenter

[1] 0

> cancor(x, y, xcenter = FALSE, ycenter = FALSE)$ycenter

[1] 0
```

partial.cor()

- **Package:** Rcmdr
- **Input:**

  X matrice di dimensione $n \times k$ le cui colonne corrispondono ai vettori numerici $x_1, x_2, \ldots, x_k$

- **Description:** 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

- **Examples:**

```
> k <- 3
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> X <- cbind(x1, x2, x3)
> X

      x1  x2   x3
[1,] 1.1 1.2 1.40
[2,] 2.3 3.4 5.60
[3,] 4.5 5.6 7.56
[4,] 6.7 7.5 6.00
[5,] 8.9 7.5 5.40
[6,] 3.4 6.7 6.60
[7,] 5.6 8.6 8.70
[8,] 6.7 7.6 8.70

> n <- 8
> R <- cor(X)
> RI <- solve(R)
> D <- 1/sqrt(diag(RI))
> mat <- -RI * (D %o% D)
> diag(mat) <- 0
> mat

           x1        x2         x3
x1  0.0000000 0.8221398 -0.4883764
x2  0.8221398 0.0000000  0.8022181
x3 -0.4883764 0.8022181  0.0000000

> partial.cor(X)

           x1        x2         x3
x1  0.0000000 0.8221398 -0.4883764
x2  0.8221398 0.0000000  0.8022181
x3 -0.4883764 0.8022181  0.0000000

> k <- 2
> x1 <- c(-1.2, -1.3, -6.7, 0.8, -7.6, -5.6)
> x2 <- c(1, 2, 3, 5, 6, 7.3)
> X <- cbind(x1, x2)
> X
```

```
       x1  x2
[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
```

```
> n <- 6
> R <- cor(X)
> RI <- solve(R)
> D <- 1/sqrt(diag(RI))
> mat <- -RI * (D %o% D)
> diag(mat) <- 0
> mat
```

```
           x1         x2
x1  0.0000000 -0.4463339
x2 -0.4463339  0.0000000
```

```
> partial.cor(X)
```

```
           x1         x2
x1  0.0000000 -0.4463339
x2 -0.4463339  0.0000000
```

cor2pcor()

- **Package:** corpcor
- **Input:**

  `m` matrice di covarianza o di correlazione di dimensione $n \times k$ dei vettori numerici $x_1, x_2, \ldots, x_k$

- **Description:** correlazione parziale
- **Formula:**

$$r_{x_i x_j|\cdot} = -\frac{R^{-1}_{i,\,j}}{\sqrt{R^{-1}_{i,\,i}\,R^{-1}_{j,\,j}}} \quad \forall i,\, j = 1,\, 2,\, \ldots,\, k$$

dove R è la matrice di correlazione tra i k vettori

- **Example 1:**

```
> k <- 3
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> X <- cbind(x1, x2, x3)
> X
```

```
      x1  x2   x3
[1,] 1.1 1.2 1.40
[2,] 2.3 3.4 5.60
[3,] 4.5 5.6 7.56
[4,] 6.7 7.5 6.00
[5,] 8.9 7.5 5.40
[6,] 3.4 6.7 6.60
[7,] 5.6 8.6 8.70
[8,] 6.7 7.6 8.70
```

```
> n <- 8
> R <- cor(X)
> RI <- solve(R)
> D <- 1/sqrt(diag(RI))
> mat <- -RI * (D %o% D)
> diag(mat) <- 1
> mat

           x1        x2         x3
x1  1.0000000 0.8221398 -0.4883764
x2  0.8221398 1.0000000  0.8022181
x3 -0.4883764 0.8022181  1.0000000

> cor2pcor(m = cor(X))

           [,1]      [,2]       [,3]
[1,]  1.0000000 0.8221398 -0.4883764
[2,]  0.8221398 1.0000000  0.8022181
[3,] -0.4883764 0.8022181  1.0000000

> cor2pcor(m = cov(X))

           [,1]      [,2]       [,3]
[1,]  1.0000000 0.8221398 -0.4883764
[2,]  0.8221398 1.0000000  0.8022181
[3,] -0.4883764 0.8022181  1.0000000
```

- **Example 2:**

```
> k <- 2
> x1 <- c(-1.2, -1.3, -6.7, 0.8, -7.6, -5.6)
> x2 <- c(1, 2, 3, 5, 6, 7.3)
> X <- cbind(x1, x2)
> X

       x1  x2
[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

> n <- 6
> R <- cor(X)
> RI <- solve(R)
> D <- 1/sqrt(diag(RI))
> mat <- -RI * (D %o% D)
> diag(mat) <- 1
> mat

           x1         x2
x1  1.0000000 -0.4463339
x2 -0.4463339  1.0000000

> cor2pcor(m = cor(X))

           [,1]       [,2]
[1,]  1.0000000 -0.4463339
[2,] -0.4463339  1.0000000
```

```
> cor2pcor(m = cov(X))

           [,1]       [,2]
[1,]  1.0000000 -0.4463339
[2,] -0.4463339  1.0000000
```

pcor2cor()

- **Package:** corpcor
- **Input:**

  m matrice di correlazione parziale di dimensione $k \times k$ dei vettori numerici $x_1, x_2, \ldots, x_k$

- **Description:** correlazione parziale
- **Formula:**

$$r_{x_i x_j} = \frac{\sigma_{x_i x_j}}{\sigma_{x_i}\,\sigma_{x_j}} = \frac{s_{x_i x_j}}{s_{x_i}\,s_{x_j}} = \frac{ss_{x_i x_j}}{\sqrt{ss_{x_i}\,ss_{x_j}}} \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 3
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> X <- cbind(x1, x2, x3)
> X

      x1  x2   x3
[1,] 1.1 1.2 1.40
[2,] 2.3 3.4 5.60
[3,] 4.5 5.6 7.56
[4,] 6.7 7.5 6.00
[5,] 8.9 7.5 5.40
[6,] 3.4 6.7 6.60
[7,] 5.6 8.6 8.70
[8,] 6.7 7.6 8.70

> n <- 8
> cor(X)

          x1        x2        x3
x1 1.0000000 0.8260355 0.5035850
x2 0.8260355 1.0000000 0.8066075
x3 0.5035850 0.8066075 1.0000000

> mat <- cor2pcor(cor(X))
> mat

           [,1]      [,2]       [,3]
[1,]  1.0000000 0.8221398 -0.4883764
[2,]  0.8221398 1.0000000  0.8022181
[3,] -0.4883764 0.8022181  1.0000000

> pcor2cor(m = mat)

          [,1]      [,2]      [,3]
[1,] 1.0000000 0.8260355 0.5035850
[2,] 0.8260355 1.0000000 0.8066075
[3,] 0.5035850 0.8066075 1.0000000
```

```
> k <- 2
> x1 <- c(-1.2, -1.3, -6.7, 0.8, -7.6, -5.6)
> x2 <- c(1, 2, 3, 5, 6, 7.3)
> X <- cbind(x1, x2)
> X

       x1  x2
[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

> n <- 6
> cor(X)

           x1         x2
x1  1.0000000 -0.4463339
x2 -0.4463339  1.0000000

> mat <- cor2pcor(m = cor(X))
> cor2pcor(m = mat)

           [,1]       [,2]
[1,]  1.0000000 -0.4463339
[2,] -0.4463339  1.0000000
```

3.14 Media e varianza pesate

weighted.mean()

- **Input:**
- **Package:** stats

  x vettore numerico di dimensione n

  w vettore numerico w di pesi di dimensione n
- **Description:** media pesata
- **Formula:**

$$\bar{x}_W = \frac{\sum_{i=1}^n x_i\, w_i}{\sum_{j=1}^n w_j}$$

- **Examples:**

```
> x <- c(3.7, 3.3, 3.5, 2.8)
> w <- c(5, 5, 4, 1)
> sum(w)

[1] 15

> sum(x * w)/sum(w)

[1] 3.453333

> weighted.mean(x, w)

[1] 3.453333
```

```
> x <- c(3.7, 3.3, 3.5, 2.8)
> w <- c(0.16, 0.34, 0.28, 0.22)
> sum(w)

[1] 1

> sum(x * w)

[1] 3.31

> weighted.mean(x, w)

[1] 3.31
```

wt.var()

- **Input:**
- **Package:** corpcor

  `xvec` vettore numerico di dimensione n

  `w` vettore numerico w di pesi a somma unitaria di dimensione n

- **Description:** varianza pesata
- **Formula:**

$$s_x^2 = (1 - w^T\, w)^{-1}\, (x - \bar{x}_W)^T\, W^{-1}\, (x - \bar{x}_W)$$

- **Examples:**

```
> x <- c(3.7, 3.3, 3.5, 2.8)
> w <- c(5, 5, 4, 1)
> w <- w/sum(w)
> xW <- sum(x * w)
> W <- diag(1/w)
> as.numeric(1/(1 - t(w) %*% w) * t(x - xW) %*% solve(W) %*% (x -
+     xW))

[1] 0.0813924

> wt.var(xvec = x, w)

[1] 0.0813924

> x <- c(3.7, 3.3, 3.5, 2.8)
> w <- c(0.16, 0.34, 0.28, 0.22)
> xW <- sum(x * w)
> W <- diag(1/w)
> as.numeric(1/(1 - t(w) %*% w) * t(x - xW) %*% solve(W) %*% (x -
+     xW))

[1] 0.1252732

> wt.var(xvec = x, w)

[1] 0.1252732
```

wt.moments()

- **Package:** corpcor
- **Input:**

  x matrice di dimensione $n \times k$ le cui colonne corrispondono ai vettori numerici $x_1, x_2, \ldots, x_k$

  w vettore numerico w di pesi a somma unitaria di dimensione n

- **Description:** media e varinza pesate pesata
- **Output:**

  mean medie pesate

  var varianze pesate

- **Formula:**

  mean

$$\bar{x}_{i\,W} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, k$$

  var

$$s^2_{x_i} \,=\, (1 - w^T\, w)^{-1}\, (x_i - \bar{x}_{i\,W})^T\, W^{-1}\, (x_i - \bar{x}_{i\,W}) \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, k$$

- **Examples 1:**

```
> k <- 2
> x1 <- c(1.2, 3.4, 5.6, 7.5, 7.7, 7.8)
> x2 <- c(1.1, 2.3, 4.4, 5.1, 2.9, 8.7)
> x <- cbind(x1, x2)
> n <- 6
> w <- c(0.16, 0.34, 0.28, 0.12, 0.08, 0.02)
> xW1 <- sum(x1 * w)
> xW2 <- sum(x2 * w)
> c(xW1, xW2)

[1] 4.588 3.208

> wt.moments(x, w)$mean

   x1    x2
4.588 3.208

> W <- diag(1/w)
> var1 <- as.numeric(1/(1 - t(w) %*% w) * t(x1 - xW1) %*% solve(W) %*%
+     (x1 - xW1))
> var2 <- as.numeric(1/(1 - t(w) %*% w) * t(x2 - xW2) %*% solve(W) %*%
+     (x2 - xW2))
> c(var1, var2)

[1] 6.061454 3.200126

> wt.moments(x, w)$var

      x1       x2
6.061454 3.200126
```

- **Examples 2:**

```
> k <- 3
> x1 <- c(1.1, 3.6, 7.4, 6.8, 9.8, 7.6, 3.8)
> x2 <- c(5.6, 7.54, 7.3, 3.5, 6.45, 5.4, 3.4)
> x3 <- c(2.8, 8.5, 6.4, 7.8, 98.6, 7.5, 5.7)
> x <- cbind(x1, x2, x3)
> n <- 7
> w <- c(0.16, 0.34, 0.15, 0.12, 0.08, 0.03, 0.12)
> xW1 <- sum(x1 * w)
> xW2 <- sum(x2 * w)
> xW3 <- sum(x3 * w)
> c(xW1, xW2, xW3)

[1]  4.7940  6.0606 14.0310

> wt.moments(x, w)$mean

     x1      x2      x3
 4.7940  6.0606 14.0310

> W <- diag(1/w)
> var1 <- as.numeric(1/(1 - t(w) %*% w) * t(x1 - xW1) %*% solve(W) %*%
+     (x1 - xW1))
> var2 <- as.numeric(1/(1 - t(w) %*% w) * t(x2 - xW2) %*% solve(W) %*%
+     (x2 - xW2))
> var3 <- as.numeric(1/(1 - t(w) %*% w) * t(x3 - xW3) %*% solve(W) %*%
+     (x3 - xW3))
> c(var1, var2, var3)

[1]   8.159415   3.336630 781.977429

> wt.moments(x, w)$var

        x1         x2         x3
  8.159415   3.336630 781.977429
```

cov.wt()

- **Package:** `stats`
- **Input:**

  `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 = TRUE / FALSE` parametro di posizione

  `cor = TRUE / FALSE` correlazione pesata

- **Description:** 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 = TRUE`

$$s_{x_i x_j} = (1 - w^T\, w)^{-1}\, (x_i - \bar{x}_{i\,W})^T\, W^{-1}\, (x_j - \bar{x}_{j\,W}) \quad \forall\, i, j = 1, 2, \ldots, k$$

`center = FALSE`

$$s_{x_i x_j} = (1 - w^T\, w)^{-1}\, x_i^T\, W^{-1}\, x_j \quad \forall\, i, j = 1, 2, \ldots, k$$

`center`

`center = TRUE`

$$\bar{x}_{i\,W} \quad \forall\, i = 1, 2, \ldots, k$$

`center = FALSE`

$$0$$

`n.obs`

$$n$$

`wt`

$$w$$

`cor`

`center = TRUE`

$$r_{x_i x_j} = \frac{(x_i - \bar{x}_{i\,W})^T\, W^{-1}\, (x_j - \bar{x}_{j\,W})}{\left((x_i - \bar{x}_{i\,W})^T\, W^{-1}\, (x_i - \bar{x}_{i\,W})\right)^{1/2}\, \left((x_j - \bar{x}_{j\,W})^T\, W^{-1}\, (x_j - \bar{x}_{j\,W})\right)^{1/2}} \quad \forall\, i, j = 1, 2, \ldots, k$$

`center = FALSE`

$$r_{x_i x_j} = \frac{x_i^T\, W^{-1}\, x_j}{\left(x_i^T\, W^{-1}\, x_i\right)^{1/2}\, \left(x_j^T\, W^{-1}\, x_j\right)^{1/2}} \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Examples 1:**

```
> k <- 2
> x1 <- c(1.2, 3.4, 5.6, 7.5, 7.7, 7.8)
> x2 <- c(1.1, 2.3, 4.4, 5.1, 2.9, 8.7)
> n <- 6
> w <- rep(1/n, times = n)
> sum(w)

[1] 1

> x1W <- sum(x1 * w)
> x2W <- sum(x2 * w)
> W <- diag(1/w)
> as.numeric(1/(1 - t(w) %*% w) * t(x1 - x1W) %*% solve(W) %*%
+     (x1 - x1W))

[1] 7.406667

> as.numeric(1/(1 - t(w) %*% w) * t(x2 - x2W) %*% solve(W) %*%
+     (x2 - x2W))

[1] 7.185667

> as.numeric(1/(1 - t(w) %*% w) * t(x1 - x1W) %*% solve(W) %*%
+     (x2 - x2W))
```

```
[1] 5.330667

> z <- cbind(x1, x2)
> cov.wt(z, wt = w, center = TRUE, cor = TRUE)$cov

         x1       x2
x1 7.406667 5.330667
x2 5.330667 7.185667

> as.numeric(1/(1 - t(w) %*% w) * t(x1) %*% solve(W) %*% x1)

[1] 44.148

> as.numeric(1/(1 - t(w) %*% w) * t(x2) %*% solve(W) %*% x2)

[1] 27.194

> as.numeric(1/(1 - t(w) %*% w) * t(x1) %*% solve(W) %*% x2)

[1] 32.444

> cov.wt(z, wt = w, center = FALSE, cor = TRUE)$cov

       x1     x2
x1 44.148 32.444
x2 32.444 27.194
```

- **Examples 2:**

```
> k <- 2
> x1 <- c(1.2, 3.4, 5.6, 7.5, 7.7, 7.8)
> x2 <- c(1.1, 2.3, 4.4, 5.1, 2.9, 8.7)
> n <- 6
> w <- rep(1/n, times = n)
> sum(w)

[1] 1

> x1W <- sum(x1 * w)
> x2W <- sum(x2 * w)
> W <- diag(1/w)
> c(x1W, x2W)

[1] 5.533333 4.083333

> cov.wt(z, wt = w, center = TRUE, cor = TRUE)$center

      x1       x2
5.533333 4.083333

> cov.wt(z, wt = w, center = FALSE, cor = TRUE)$center

[1] 0
```

- **Examples 3:**

```
> k <- 2
> x1 <- c(1.2, 3.4, 5.6, 7.5, 7.7, 7.8)
> x2 <- c(1.1, 2.3, 4.4, 5.1, 2.9, 8.7)
> n <- 6
> w <- rep(1/n, times = n)
> sum(w)

[1] 1

> n

[1] 6

> cov.wt(z, wt = w, center = TRUE, cor = TRUE)$n.obs

[1] 6

> cov.wt(z, wt = w, center = FALSE, cor = TRUE)$n.obs

[1] 6
```

- **Example 4:**

```
> k <- 2
> x1 <- c(1.2, 3.4, 5.6, 7.5, 7.7, 7.8)
> x2 <- c(1.1, 2.3, 4.4, 5.1, 2.9, 8.7)
> n <- 6
> w <- rep(1/n, times = n)
> sum(w)

[1] 1

> w

[1] 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667

> cov.wt(z, wt = w, center = TRUE, cor = TRUE)$wt

[1] 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667

> cov.wt(z, wt = w, center = FALSE, cor = TRUE)$wt

[1] 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667 0.1666667
```

- **Example 5:**

```
> k <- 2
> x1 <- c(1.2, 3.4, 5.6, 7.5, 7.7, 7.8)
> x2 <- c(1.1, 2.3, 4.4, 5.1, 2.9, 8.7)
> n <- 6
> w <- rep(1/n, times = n)
> sum(w)

[1] 1

> x1W <- sum(x1 * w)
> x2W <- sum(x2 * w)
> W <- diag(1/w)
> covx1x2 <- 1/(1 - t(w) %*% w) * t(x1 - x1W) %*% solve(W) %*%
+     (x2 - x2W)
> covx1x2 <- as.numeric(covx1x2)
> covx1x2
```

```
[1] 5.330667

> sx1 <- sqrt(1/(1 - t(w) %*% w) * t(x1 - x1W) %*% solve(W) %*%
+     (x1 - x1W))
> sx1 <- as.numeric(sx1)
> sx1

[1] 2.721519

> sx2 <- sqrt(1/(1 - t(w) %*% w) * t(x2 - x2W) %*% solve(W) %*%
+     (x2 - x2W))
> sx2 <- as.numeric(sx2)
> sx2

[1] 2.680609

> rx1x2 <- covx1x2/(sx1 * sx2)
> rx1x2

[1] 0.7306958

> cov.wt(z, wt = w, center = TRUE, cor = TRUE)$cor

          x1        x2
x1 1.0000000 0.7306958
x2 0.7306958 1.0000000

> covx1x2 <- as.numeric(1/(1 - t(w) %*% w) * t(x1) %*% solve(W) %*%
+     x2)
> covx1x2

[1] 32.444

> sx1 <- sqrt(as.numeric(1/(1 - t(w) %*% w) * t(x1) %*% solve(W) %*%
+     x1))
> sx1

[1] 6.644396

> sx2 <- sqrt(as.numeric(1/(1 - t(w) %*% w) * t(x2) %*% solve(W) %*%
+     x2))
> sx2

[1] 5.214787

> rx1x2 <- covx1x2/(sx1 * sx2)
> rx1x2

[1] 0.9363589

> cov.wt(z, wt = w, center = FALSE, cor = TRUE)$cor

          x1        x2
x1 1.0000000 0.9363589
x2 0.9363589 1.0000000
```

- **Example 6:**

```
> k <- 3
> x1 <- c(1.1, 3.6, 7.4, 6.8, 9.8, 7.6, 3.8)
> x2 <- c(5.6, 7.54, 7.3, 3.5, 6.45, 5.4, 3.4)
> x3 <- c(2.8, 8.5, 6.4, 7.8, 98.6, 7.5, 5.7)
> n <- 7
> w <- rep(1/n, times = n)
> sum(w)

[1] 1

> x1W <- sum(x1 * w)
> x2W <- sum(x2 * w)
> x3W <- sum(x3 * w)
> W <- diag(1/w)
> as.numeric(1/(1 - t(w) %*% w) * t(x1 - x1W) %*% solve(W) %*%
+     (x1 - x1W))

[1] 8.949048

> as.numeric(1/(1 - t(w) %*% w) * t(x2 - x2W) %*% solve(W) %*%
+     (x2 - x2W))

[1] 2.777681

> as.numeric(1/(1 - t(w) %*% w) * t(x3 - x3W) %*% solve(W) %*%
+     (x3 - x3W))

[1] 1216.591

> as.numeric(1/(1 - t(w) %*% w) * t(x1 - x1W) %*% solve(W) %*%
+     (x2 - x2W))

[1] 0.631881

> as.numeric(1/(1 - t(w) %*% w) * t(x1 - x1W) %*% solve(W) %*%
+     (x3 - x3W))

[1] 65.41452

> as.numeric(1/(1 - t(w) %*% w) * t(x2 - x2W) %*% solve(W) %*%
+     (x3 - x3W))

[1] 13.49269

> z <- cbind(x1, x2, x3)
> cov.wt(z, wt = w, center = TRUE, cor = TRUE)$cov

          x1        x2         x3
x1  8.949048  0.631881   65.41452
x2  0.631881  2.777681   13.49269
x3 65.414524 13.492690 1216.59143

> as.numeric(1/(1 - t(w) %*% w) * t(x1) %*% solve(W) %*% x1)

[1] 47.235

> as.numeric(1/(1 - t(w) %*% w) * t(x2) %*% solve(W) %*% x2)

[1] 39.34568
```

```
> as.numeric(1/(1 - t(w) %*% w) * t(x3) %*% solve(W) %*% x3)
```

```
[1] 1665.432
```

```
> as.numeric(1/(1 - t(w) %*% w) * t(x1) %*% solve(W) %*% x2)
```

```
[1] 38.049
```

```
> as.numeric(1/(1 - t(w) %*% w) * t(x1) %*% solve(W) %*% x3)
```

```
[1] 196.5033
```

```
> as.numeric(1/(1 - t(w) %*% w) * t(x2) %*% solve(W) %*% x3)
```

```
[1] 141.6067
```

```
> cov.wt(z, wt = w, center = FALSE, cor = TRUE)$cov
```

```
         x1        x2        x3
x1  47.2350  38.04900  196.5033
x2  38.0490  39.34568  141.6067
x3 196.5033 141.60667 1665.4317
```

- **Example 7:**

```
> k <- 3
> x1 <- c(1.1, 3.6, 7.4, 6.8, 9.8, 7.6, 3.8)
> x2 <- c(5.6, 7.54, 7.3, 3.5, 6.45, 5.4, 3.4)
> x3 <- c(2.8, 8.5, 6.4, 7.8, 98.6, 7.5, 5.7)
> n <- 7
> w <- rep(1/n, times = n)
> sum(w)
```

```
[1] 1
```

```
> c(x1W, x2W, x3W)
```

```
[1]  5.728571  5.598571 19.614286
```

```
> cov.wt(z, wt = w, center = TRUE, cor = TRUE)$center
```

```
       x1        x2        x3
 5.728571  5.598571 19.614286
```

```
> cov.wt(z, wt = w, center = FALSE, cor = TRUE)$center
```

```
[1] 0
```

- **Example 8:**

```
> k <- 3
> x1 <- c(1.1, 3.6, 7.4, 6.8, 9.8, 7.6, 3.8)
> x2 <- c(5.6, 7.54, 7.3, 3.5, 6.45, 5.4, 3.4)
> x3 <- c(2.8, 8.5, 6.4, 7.8, 98.6, 7.5, 5.7)
> n <- 7
> w <- rep(1/n, times = n)
> sum(w)
```

```
[1] 1
```

```
> n

[1] 7

> cov.wt(z, wt = w, center = TRUE, cor = TRUE)$n.obs

[1] 7

> cov.wt(z, wt = w, center = FALSE, cor = TRUE)$n.obs

[1] 7
```

- **Example 9:**

```
> k <- 3
> x1 <- c(1.1, 3.6, 7.4, 6.8, 9.8, 7.6, 3.8)
> x2 <- c(5.6, 7.54, 7.3, 3.5, 6.45, 5.4, 3.4)
> x3 <- c(2.8, 8.5, 6.4, 7.8, 98.6, 7.5, 5.7)
> n <- 7
> w <- rep(1/n, times = n)
> sum(w)

[1] 1

> w

[1] 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571

> cov.wt(z, wt = w, center = TRUE, cor = TRUE)$wt

[1] 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571

> cov.wt(z, wt = w, center = FALSE, cor = TRUE)$wt

[1] 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571
```

- **Example 10:**

```
> k <- 3
> x1 <- c(1.1, 3.6, 7.4, 6.8, 9.8, 7.6, 3.8)
> x2 <- c(5.6, 7.54, 7.3, 3.5, 6.45, 5.4, 3.4)
> x3 <- c(2.8, 8.5, 6.4, 7.8, 98.6, 7.5, 5.7)
> n <- 7
> w <- rep(1/n, times = n)
> sum(w)

[1] 1

> x1W <- sum(x1 * w)
> x2W <- sum(x2 * w)
> x3W <- sum(x3 * w)
> W <- diag(1/w)
> covx1x2 <- 1/(1 - t(w) %*% w) * t(x1 - x1W) %*% solve(W) %*%
+     (x2 - x2W)
> covx1x2 <- as.numeric(covx1x2)
> covx1x2

[1] 0.631881
```

```
> covx1x3 <- 1/(1 - t(w) %*% w) * t(x1 - x1W) %*% solve(W) %*%
+     (x3 - x3W)
> covx1x3 <- as.numeric(covx1x3)
> covx1x3

[1] 65.41452

> covx2x3 <- 1/(1 - t(w) %*% w) * t(x2 - x2W) %*% solve(W) %*%
+     (x3 - x3W)
> covx2x3 <- as.numeric(covx2x3)
> covx2x3

[1] 13.49269

> sx1 <- sqrt(1/(1 - t(w) %*% w) * t(x1 - x1W) %*% solve(W) %*%
+     (x1 - x1W))
> sx1 <- as.numeric(sx1)
> sx1

[1] 2.991496

> sx2 <- sqrt(1/(1 - t(w) %*% w) * t(x2 - x2W) %*% solve(W) %*%
+     (x2 - x2W))
> sx2 <- as.numeric(sx2)
> sx2

[1] 1.666638

> sx3 <- sqrt(1/(1 - t(w) %*% w) * t(x3 - x3W) %*% solve(W) %*%
+     (x3 - x3W))
> sx3 <- as.numeric(sx3)
> sx3

[1] 34.87967

> rx1x2 <- covx1x2/(sx1 * sx2)
> rx1x2

[1] 0.1267377

> rx1x3 <- covx1x3/(sx1 * sx3)
> rx1x3

[1] 0.6269218

> rx2x3 <- covx2x3/(sx2 * sx3)
> rx2x3

[1] 0.2321053

> cov.wt(z, wt = w, center = TRUE, cor = TRUE)$cor

          x1        x2        x3
x1 1.0000000 0.1267377 0.6269218
x2 0.1267377 1.0000000 0.2321053
x3 0.6269218 0.2321053 1.0000000

> covx1x2 <- as.numeric(1/(1 - t(w) %*% w) * t(x1) %*% solve(W) %*%
+     x2)
> covx1x2
```

```
[1] 38.049

> covx1x3 <- as.numeric(1/(1 - t(w) %*% w) * t(x1) %*% solve(W) %*%
+     x3)
> covx1x3

[1] 196.5033

> covx2x3 <- as.numeric(1/(1 - t(w) %*% w) * t(x2) %*% solve(W) %*%
+     x3)
> covx2x3

[1] 141.6067

> sx1 <- sqrt(as.numeric(1/(1 - t(w) %*% w) * t(x1 - x1W) %*% solve(W) %*%
+     (x1 - x1W)))
> sx1

[1] 2.991496

> sx1 <- sqrt(as.numeric(1/(1 - t(w) %*% w) * t(x1) %*% solve(W) %*%
+     x1))
> sx1

[1] 6.872772

> sx2 <- sqrt(as.numeric(1/(1 - t(w) %*% w) * t(x2) %*% solve(W) %*%
+     x2))
> sx2

[1] 6.272614

> sx3 <- sqrt(as.numeric(1/(1 - t(w) %*% w) * t(x3) %*% solve(W) %*%
+     x3))
> sx3

[1] 40.8097

> rx1x2 <- covx1x2/(sx1 * sx2)
> rx1x2

[1] 0.8825976

> rx1x3 <- covx1x3/(sx1 * sx3)
> rx1x3

[1] 0.7006071

> rx2x3 <- covx2x3/(sx2 * sx3)
> rx2x3

[1] 0.5531867

> cov.wt(z, wt = w, center = FALSE, cor = TRUE)$cor

          x1        x2        x3
x1 1.0000000 0.8825976 0.7006071
x2 0.8825976 1.0000000 0.5531867
x3 0.7006071 0.5531867 1.0000000
```

- **Note 1:** W è la matrice diagonale definita positiva di dimensione $n \times n$ tale che $W = \text{diag}(w_1^{-1}, w_2^{-1}, \ldots, w_n^{-1})$
- **Note 2:** Naturalmente vale che $s_{x_i x_i} = s_{x_i}^2 \quad \forall\, i = 1, 2, \ldots, k$.

corr()

- **Package:** boot
- **Input:**

  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

- **Description:** correlazione pesata
- **Formula:**

$$r_{xy} = \frac{(x - \bar{x}_W)^T \, W^{-1} \, (y - \bar{y}_W)}{\left((x - \bar{x}_W)^T \, W^{-1} \, (x - \bar{x}_W)\right)^{1/2} \left((y - \bar{y}_W)^T \, W^{-1} \, (y - \bar{y}_W)\right)^{1/2}}$$

- **Examples:**

```
> x <- c(1.2, 2.3, 3.4, 4.5, 5.6, 6.7)
> y <- c(1, 2, 3, 5, 6, 7.3)
> d <- as.matrix(cbind(x, y))
> n <- 6
> w <- abs(rnorm(n))
> w <- w/sum(w)
> sum(w)

[1] 1

> mxw <- weighted.mean(x, w)
> myw <- weighted.mean(y, w)
> W <- diag(1/w)
> num <- as.numeric(t(x - mxw) %*% solve(W) %*% (y - myw))
> den <- as.numeric(sqrt(t(x - mxw) %*% solve(W) %*% (x - mxw) *
+     t(y - myw) %*% solve(W) %*% (y - myw)))
> rho <- num/den
> rho

[1] 0.9988987

> corr(d, w)

[1] 0.9988987

> x <- c(1, 2, 3, 5.6, 7.6, 2.3, 1)
> y <- c(1.2, 2.2, 3, 15.6, 71.6, 2.2, 1.2)
> d <- as.matrix(cbind(x, y))
> n <- 7
> w <- abs(rnorm(n))
> w <- w/sum(w)
> sum(w)

[1] 1

> mxw <- weighted.mean(x, w)
> myw <- weighted.mean(y, w)
> W <- diag(1/w)
> num <- as.numeric(t(x - mxw) %*% solve(W) %*% (y - myw))
> den <- as.numeric(sqrt(t(x - mxw) %*% solve(W) %*% (x - mxw) *
+     t(y - myw) %*% solve(W) %*% (y - myw)))
> rho <- num/den
> rho

[1] 0.9095326
```

```
> corr(d, w)

[1] 0.9095326

> x <- c(1.1, 2.3, 4.5, 6.7, 8.9)
> y <- c(2.3, 4.5, 6.7, 8.9, 10.2)
> d <- as.matrix(cbind(x, y))
> n <- 5
> w <- rep(1/n, times = n)
> sum(w)

[1] 1

> mxw <- weighted.mean(x, w)
> myw <- weighted.mean(y, w)
> W <- diag(1/w)
> num <- as.numeric(t(x - mxw) %*% solve(W) %*% (y - myw))
> den <- as.numeric(sqrt(t(x - mxw) %*% solve(W) %*% (x - mxw) *
+     t(y - myw) %*% solve(W) %*% (y - myw)))
> rho <- num/den
> rho

[1] 0.9866942

> corr(d, w)

[1] 0.9866942
```

- **Note:** W è la matrice diagonale definita positiva di dimensione $n \times n$ tale che $W = \text{diag}(w_1^{-1}, w_2^{-1}, \ldots, w_n^{-1})$

3.15 Momenti centrati e non centrati

moment()

- **Package:** moments
- **Input:**

  `x` vettore numerico di dimensione n

  `order` il valore k dell'ordine

  `central = TRUE / FALSE` parametro di posizione

  `absolute = TRUE / FALSE` modulo

- **Description:** momento centrato e non centrato di ordine k
- **Formula:**

	absolute = TRUE	absolute = FALSE
central = TRUE	$\sum_{i=1}^{n} \lvert x_i - \bar{x} \rvert^k \,/\, n$	$\sum_{i=1}^{n} (x_i - \bar{x})^k \,/\, n$
central = FALSE	$\sum_{i=1}^{n} \lvert x_i \rvert^k \,/\, n$	$\sum_{i=1}^{n} x_i^k \,/\, n$

- **Examples:**

```
> x <- c(-1.2, 1.2, 3.4, 4.2, 12.4, 13.4, 17.3, 18.1)
> n <- 8
> k <- 5
> mean(abs(x - mean(x))^k)
```

```
[1] 31074.24

> moment(x, central = TRUE, absolute = TRUE, order = 5)

[1] 31074.24

> mean((x - mean(x))^k)

[1] 1565.904

> moment(x, central = TRUE, absolute = FALSE, order = 5)

[1] 1565.904

> mean(abs(x)^k)

[1] 527406.3

> moment(x, central = FALSE, absolute = TRUE, order = 5)

[1] 527406.3

> mean(x^k)

[1] 527405.6

> moment(x, central = FALSE, absolute = FALSE, order = 5)

[1] 527405.6

> x <- c(1.2, 4.5, 6.7, 7.8, 9.8)
> n <- 5
> k <- 3
> mean(abs(x - mean(x))^k)

[1] 35.0028

> moment(x, central = TRUE, absolute = TRUE, order = 3)

[1] 35.0028

> mean((x - mean(x))^k)

[1] -10.584

> moment(x, central = TRUE, absolute = FALSE, order = 3)

[1] -10.584

> mean(abs(x)^k)

[1] 361.872

> moment(x, central = FALSE, absolute = TRUE, order = 3)

[1] 361.872

> mean(x^k)

[1] 361.872

> moment(x, central = FALSE, absolute = FALSE, order = 3)

[1] 361.872
```

scale()

- **Package:** base
- **Input:**

  `x` vettore numerico di dimensione n

  `center = TRUE / FALSE` parametro di posizione

  `scale = TRUE / FALSE` parametro di scala

- **Description:** centratura o normalizzazione
- **Formula:**

	scale = TRUE	scale = FALSE
center = TRUE	$(x - \bar{x}) / s_x$	$x - \bar{x}$
center = FALSE	$x \, / \left(\frac{1}{n-1} \sum_{i=1}^{n} x_i^2 \right)^{1/2}$	x

- **Examples:**

```
> x <- c(1.2, 3.4, 4.2, 12.4, 13.4, 17.3, 18.1)
> n <- 7
> (x - mean(x))/sd(x)

[1] -1.2639104 -0.9479328 -0.8330319  0.3447028  0.4883290  1.0484712  1.1633721

> as.numeric(scale(x, center = TRUE, scale = TRUE))

[1] -1.2639104 -0.9479328 -0.8330319  0.3447028  0.4883290  1.0484712  1.1633721

> x - mean(x)

[1] -8.8 -6.6 -5.8  2.4  3.4  7.3  8.1

> as.numeric(scale(x, center = TRUE, scale = FALSE))

[1] -8.8 -6.6 -5.8  2.4  3.4  7.3  8.1

> x/sqrt(sum(x^2)/(n - 1))

[1] 0.09337932 0.26457475 0.32682763 0.96491968 1.04273578 1.34621858 1.40847146

> as.numeric(scale(x, center = FALSE, scale = TRUE))

[1] 0.09337932 0.26457475 0.32682763 0.96491968 1.04273578 1.34621858 1.40847146

> x <- c(1.2, 3.4, 4.2, 12.4, 13.4, 17.3, 18.1)
> as.numeric(scale(x, center = FALSE, scale = FALSE))

[1]  1.2  3.4  4.2 12.4 13.4 17.3 18.1

> x <- c(1.2, 4.5, 6.7, 7.8, 9.8)
> n <- 5
> (x - mean(x))/sd(x)

[1] -1.4562179 -0.4550681  0.2123651  0.5460817  1.1528392

> as.numeric(scale(x, center = TRUE, scale = TRUE))
```

```
[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 = TRUE, scale = FALSE))

[1] -4.8 -1.5  0.7  1.8  3.8

> x/sqrt(sum(x^2)/(n - 1))

[1] 0.1605504 0.6020639 0.8964063 1.0435775 1.3111615

> as.numeric(scale(x, center = FALSE, scale = TRUE))

[1] 0.1605504 0.6020639 0.8964063 1.0435775 1.3111615

> x <- c(1.2, 4.5, 6.7, 7.8, 9.8)
> as.numeric(scale(x, center = FALSE, scale = FALSE))

[1] 1.2 4.5 6.7 7.8 9.8
```

cum3()

- **Package:** boot
- **Input:**

  a vettore numerico x di dimensione n

  b vettore numerico y di dimensione n

  c vettore numerico z di dimensione n

  unbiased = TRUE / FALSE distorsione
- **Description:** momento terzo centrato
- **Formula:**

unbiased = TRUE

$$\frac{n}{(n-1)\,(n-2)} \sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y})\,(z_i - \bar{z})$$

unbiased = FALSE

$$\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})\,(y_i - \bar{y})\,(z_i - \bar{z})$$

- **Examples:**

```
> x <- c(-3, -2, -1, 0, 1, 2)
> y <- c(1.2, 2.3, 2, 3.1, 3.55, 6.7)
> z <- c(2, 3.45, 2.6, 3.11, 3.5, 6.2)
> n <- 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 = TRUE)
```

```
[1] 4.96385
```

```
> x <- c(-3, -2, -1, 0, 1, 2)
> y <- c(1.2, 2.3, 2, 3.1, 3.55, 6.7)
> z <- c(2, 3.45, 2.6, 3.11, 3.5, 6.2)
> n <- 6
> (1/n) * sum((x - mean(x)) * (y - mean(y)) * (z - mean(z)))
```

```
[1] 2.757694
```

```
> cum3(a = x, b = y, c = z, unbiased = FALSE)
```

```
[1] 2.757694
```

emm()

- **Package:** actuar
- **Input:**

  x vettore numerico di dimensione n

  order il valore k dell'ordine
- **Description:** momento non centrato di ordine k
- **Formula:**

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

- **Examples:**

```
> x <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> k <- 3
> mean(x^3)
```

```
[1] 534.2372
```

```
> emm(x, order = 3)
```

```
[1] 534.2372
```

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> n <- 5
> k <- 4
> mean(x^4)
```

```
[1] 1745.677
```

```
> emm(x, order = 4)
```

```
[1] 1745.677
```

3.16 Connessione e dipendenza in media

eta()

- **Package:** labstatR
- **Input:**

  y vettore numerico di dimensione n

  f fattore a k livelli di dimensione n

- **Description:** rapporto di correlazione $\eta^2_{y|f}$
- **Formula:**

$$\eta^2_{y|f} = \frac{\sum_{j=1}^{k} (\bar{y}_j - \bar{y})^2 \, n_j}{\sum_{i=1}^{n} (\bar{y}_i - \bar{y})^2}$$

- **Examples:**

```
> y <- c(1, 1.2, 2.1, 3.4, 5.4, 5.6, 7.2, 3.2, 3, 1, 2.3)
> f <- factor(c("a", "b", "c", "b", "a", "c", "a", "b", "b", "c",
+     "a"))
> f

 [1] a b c b a c a b b c a
Levels: a b c

> k <- 3
> n <- 11
> table(f)

f
a b c
4 4 3

> enne <- tapply(y, f, FUN = length)
> enne

a b c
4 4 3

> ymedio <- tapply(y, f, FUN = mean)
> sum((ymedio - mean(y))^2 * enne)/sum((y - mean(y))^2)

[1] 0.08657807

> eta(f, y)

[1] 0.08657807

> y <- c(1.2, 3.4, 55.6, 5.1, 7.8, 8.4, 8.7, 9.8)
> f <- factor(c("a", "b", "b", "b", "b", "a", "a", "b"))
> f

[1] a b b b b a a b
Levels: a b

> k <- 2
> n <- 8
> table(f)
```

```
f
a b
3 5

> enne <- tapply(y, f, FUN = length)
> enne

a b
3 5

> ymedio <- tapply(y, f, FUN = mean)
> sum((ymedio - mean(y))^2 * enne)/sum((y - mean(y))^2)

[1] 0.0900426

> eta(f, y)

[1] 0.0900426
```

Gini()

- **Package:** ineq
- **Input:**

  x vettore numerico di dimensione n
- **Description:** rapporto di concentrazione di *Gini*
- **Formula:**

$$\frac{n-1}{n}\,G$$

$$\text{dove} \quad G = \frac{2}{n-1}\sum_{i=1}^{n-1}(p_i - q_i) = 1 - \frac{2}{n-1}\sum_{i=1}^{n-1} q_i = \frac{1}{n\,(n-1)\,\bar{x}}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left(x_{(j)} - x_{(i)}\right)$$

- **Examples:**

```
> x <- c(1, 1, 1, 4, 4, 5, 7, 10)
> x <- sort(x)
> x

[1]  1  1  1  4  4  5  7 10

> n <- 8
> q <- cumsum(x[1:(n - 1)])/sum(x)
> G <- 2/(n - 1) * sum((1:(n - 1))/n - q)
> G

[1] 0.4545455

> R <- (n - 1)/n * G
> R

[1] 0.3977273

> Gini(x)

[1] 0.3977273
```

```
> x <- c(1.2, 3.4, 55.6, 5.1, 7.8, 8.4, 8.7, 9.8)
> x <- sort(x)
> x

[1]  1.2  3.4  5.1  7.8  8.4  8.7  9.8 55.6

> n <- 8
> q <- cumsum(x[1:(n - 1)])/sum(x)
> G <- 2/(n - 1) * sum((1:(n - 1))/n - q)
> G

[1] 0.606

> R <- (n - 1)/n * G
> R

[1] 0.53025

> Gini(x)

[1] 0.53025
```

gini()

- **Package:** labstatR
- **Input:**

  y vettore numerico di dimensione n

  plot = FALSE
- **Description:** indici di concentrazione
- **Output:**

  G indice di *Gini*

  R rapporto di concentrazione di *Gini*

  P proporzioni

  Q somme cumulate
- **Formula:**

  G

$$G = \frac{2}{n-1} \sum_{i=1}^{n-1} (p_i - q_i) = 1 - \frac{2}{n-1} \sum_{i=1}^{n-1} q_i = \frac{1}{n\,(n-1)\,\bar{y}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \left(y_{(j)} - y_{(i)}\right)$$

$$\text{dove} \qquad p_i = i \,/\, n \quad \forall\, i = 1, 2, \ldots, n$$

$$q_i = \textstyle\sum_{j=1}^{i} y_{(j)} \,/\, \sum_{j=1}^{n} y_j \quad \forall\, i = 1, 2, \ldots, n$$

  R

$$\frac{n-1}{n}\, G$$

  P

$$0,\, p_i \quad \forall\, i = 1, 2, \ldots, n$$

  Q

$$0,\, q_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> y <- c(1, 1, 1, 4, 4, 5, 7, 10)
> y <- sort(y)
> y

[1]  1  1  1  4  4  5  7 10

> n <- 8
> q <- cumsum(y[1:(n - 1)])/sum(y)
> G <- 2/(n - 1) * sum((1:(n - 1))/n - q)
> G

[1] 0.4545455

> gini(y, plot = FALSE)$G

[1] 0.4545455

> R <- (n - 1)/n * G
> R

[1] 0.3977273

> gini(y, plot = FALSE)$R

[1] 0.3977273

> 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 = FALSE)$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] 0.00000000 0.03030303 0.06060606 0.09090909 0.21212121 0.33333333 0.48484848
[8] 0.69696970 1.00000000

> gini(y, plot = FALSE)$Q

[1] 0.00000000 0.03030303 0.06060606 0.09090909 0.21212121 0.33333333 0.48484848
[8] 0.69696970 1.00000000

> y <- c(1.2, 3.4, 55.6, 5.1, 7.8, 8.4, 8.7, 9.8)
> y <- sort(y)
> y

[1]  1.2  3.4  5.1  7.8  8.4  8.7  9.8 55.6

> n <- 8
> q <- cumsum(y[1:(n - 1)])/sum(y)
> G <- 2/(n - 1) * sum((1:(n - 1))/n - q)
> G

[1] 0.606
```

```
> gini(y, plot = FALSE)$G

[1] 0.606

> R <- (n - 1)/n * G
> R

[1] 0.53025

> gini(y, plot = FALSE)$R

[1] 0.53025

> 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 = FALSE)$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] 0.000 0.012 0.046 0.097 0.175 0.259 0.346 0.444 1.000

> gini(y, plot = FALSE)$Q

[1] 0.000 0.012 0.046 0.097 0.175 0.259 0.346 0.444 1.000
```

RS()

- **Package:** ineq
- **Input:**

  x vettore numerico di dimensione n
- **Description:** coefficiente di disuguaglianza di *Ricci - Schutz*
- **Formula:**

$$\frac{1}{2\,n\,\bar{x}} \sum_{i=1}^{n} |\,x_i - \bar{x}\,|$$

- **Examples:**

```
> x <- c(1, 1.2, 3.4, 0.8)
> mean(abs(x - mean(x)))/(2 * mean(x))

[1] 0.28125

> RS(x)

[1] 0.28125

> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> mean(abs(x - mean(x)))/(2 * mean(x))

[1] 0.1417790

> RS(x)

[1] 0.1417790
```

chi2()

- **Package:** labstatR
- **Input:**

  f fattore a k livelli

  g fattore a h livelli

- **Description:** quadrato dell'indice di connessione $\tilde{\chi}^2$ di *Cramer*
- **Formula:**

$$\tilde{\chi}^2 = \frac{\chi^2}{\chi^2_{\max}} = \frac{\sum_{i=1}^{k} \sum_{j=1}^{h} \frac{(n_{ij} - \hat{n}_{ij})^2}{\hat{n}_{ij}}}{n_{\cdot\cdot} \, \min(k-1, \, h-1)} = \frac{\sum_{i=1}^{h} \sum_{j=1}^{k} \frac{n_{ij}^2}{\hat{n}_{ij}} - n_{\cdot\cdot}}{n_{\cdot\cdot} \, \min(k-1, \, h-1)} = \frac{\sum_{i=1}^{k} \sum_{j=1}^{h} \frac{n_{ij}^2}{n_{i\cdot} \, n_{\cdot j}} - 1}{\min(k-1, \, h-1)}$$

$$\text{dove} \quad \hat{n}_{ij} = \frac{n_{i\cdot} \, n_{\cdot j}}{n_{\cdot\cdot}} \quad \forall\, i = 1, 2, \dots, k \quad \forall\, j = 1, 2, \dots, h$$

$$n_{\cdot\cdot} = \sum_{i=1}^{k} \sum_{j=1}^{h} n_{ij} = \sum_{i=1}^{k} \sum_{j=1}^{h} \hat{n}_{ij}$$

- **Examples:**

```
> f <- factor(c("a", "b", "c", "b", "a", "c", "a", "b", "b", "c",
+     "a"))
> f

 [1] a b c b a c a b b c a
Levels: a b c

> k <- nlevels(f)
> g <- factor(c("O", "P", "W", "P", "P", "O", "O", "W", "W", "P",
+     "P"))
> g

 [1] O P W P P O O W W P P
Levels: O P W

> h <- nlevels(g)
> 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))
> chi2(f, g)

[1] 0.1777778

> f <- factor(c("a", "b", "b", "b", "b", "a", "a", "b"))
> f

[1] a b b b b a a b
Levels: a b

> k <- nlevels(f)
> g <- factor(c("A", "B", "B", "B", "A", "A", "B", "A"))
> g
```

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

> h <- nlevels(g)
> table(f, g)

   g
f   A B
  a 2 1
  b 2 3

> n.. <- sum(table(f, g))
> chi2(f, g)

[1] 0.06666667
```

E()

- **Package:** labstatR
- **Input:**

  f fattore a k livelli di dimensione n

- **Description:** indice di eterogeneità di *Gini*
- **Formula:**

$$E = \frac{k}{k-1} \left(1 - \frac{1}{n^2} \sum_{i=1}^{k} n_i^2 \right)$$

- **Examples:**

```
> f <- factor(c("a", "b", "c", "b", "a", "c", "a", "b", "b", "c",
+     "a"))
> f

 [1] a b c b a c a b b c a
Levels: a b c

> k <- 3
> n <- 11
> enne <- table(f)
> enne

f
a b c
4 4 3

> E <- k/(k - 1) * (1 - 1/n^2 * sum(enne^2))
> E

[1] 0.9917355

> E(f)

[1] 0.9917355

> f <- factor(c("A", "B", "B", "B", "A", "A", "B", "A"))
> f
```

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

> k <- 2
> n <- 8
> enne <- table(f)
> enne

f
A B
4 4

> E <- k/(k - 1) * (1 - 1/n^2 * sum(enne^2))
> E

[1] 1

> E(g)

[1] 1
```

3.17 Sintesi di dati

summary()

- **Package:** `base`
- **Input:**

  `x` vettore numerico di dimensione n

- **Description:** statistiche descrittive
- **Output:**

  `Min.` minimo

  `1st Qu.` primo quartile

  `Median` mediana

  `Mean` media aritmetica

  `3rd Qu.` terzo quartile

  `Max.` massimo

- **Formula:**

  `Min.`

  $$x_{(1)}$$

  `1st Qu.`

  $$Q_{0.25}(x)$$

  `Median`

  $$Q_{0.5}(x)$$

  `Mean`

  $$\bar{x}$$

  `3rd Qu.`

  $$Q_{0.75}(x)$$

  `Max.`

  $$x_{(n)}$$

- **Examples:**

```
> x <- c(1, 2.3, 5, 6.7, 8)
> 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 <- c(1.2, 2.2, 3, 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
```

- **Note:** Calcola i quartili con la funzione `quantile()`.

fivenum()

- **Package:** stats
- **Input:**

  x vettore numerico di dimensione n

- **Description:** cinque numeri di *Tukey*
- **Formula:**

$$x_{(1)}$$

$$0.5\,\left(x_{\lfloor\,\lfloor\,(n+3)\,/\,2\rfloor\,/\,2\rfloor} + x_{\lceil\,\lfloor\,(n+3)\,/\,2\rfloor\,/\,2\rceil}\right)$$

$$Q_{0.5}(x)$$

$$0.5\,\left(x_{\lfloor\,n+1-\lfloor\,(n+3)\,/\,2\rfloor\,/\,2\rfloor} + x_{\lceil\,n+1-\lfloor\,(n+3)\,/\,2\rfloor\,/\,2\rceil}\right)$$

$$x_{(n)}$$

- **Examples:**

```
> x <- c(1, 2.3, 5, 6.7, 8)
> n <- 5
> min(x)

[1] 1

> 0.5 * (x[floor(floor((n + 3)/2)/2)] + x[ceiling(floor((n + 3)/2)/2)])

[1] 2.3

> median(x)

[1] 5

> 0.5 * (x[n + 1 - floor(floor((n + 3)/2)/2)] + x[n + 1 - ceiling(floor((n +
+     3)/2)/2)])

[1] 6.7

> max(x)

[1] 8

> fivenum(x)

[1] 1.0 2.3 5.0 6.7 8.0

> x <- c(1.2, 1.2, 2.2, 2.2, 3, 15.6, 71.6)
> n <- 7
> min(x)

[1] 1.2

> 0.5 * (x[floor(floor((n + 3)/2)/2)] + x[ceiling(floor((n + 3)/2)/2)])

[1] 1.7
```

```
> median(x)

[1] 2.2

> 0.5 * (x[n + 1 - floor(floor((n + 3)/2)/2)] + x[n + 1 - ceiling(floor((n +
+     3)/2)/2)])

[1] 9.3

> max(x)

[1] 71.6

> fivenum(x)

[1]  1.2  1.7  2.2  9.3 71.6

> x <- c(1.44, 5.76, 21.16, 60.84)
> n <- 4
> min(x)

[1] 1.44

> 0.5 * (x[floor(floor((n + 3)/2)/2)] + x[ceiling(floor((n + 3)/2)/2)])

[1] 3.6

> median(x)

[1] 13.46

> 0.5 * (x[n + 1 - floor(floor((n + 3)/2)/2)] + x[n + 1 - ceiling(floor((n +
+     3)/2)/2)])

[1] 41

> max(x)

[1] 60.84

> fivenum(x)

[1]  1.44  3.60 13.46 41.00 60.84
```

basicStats()

- **Package:** fBasics
- **Input:**

  `x` vettore numerico di dimensione n

  `ci` livello di confidenza $1 - \alpha$

- **Description:** statistiche riassuntive
- **Output:**

  `nobs` dimensione campionaria

  `NAs` numero di valori `NA` oppure `NaN`

  `Minimum` minimo

  `Maximum` massimo

  `1. Quartile` primo quartile

  `3. Quartile` terzo quartile

  `Mean` media aritmetica

  `Median` mediana

  `Sum` somma

  `SE Mean` errore standard della media

  `LCL Mean` estremo inferiore dell'intervallo di confidenza a livello $1 - \alpha$ per la media incognita

  `UCL Mean` estremo superiore dell'intervallo di confidenza a livello $1 - \alpha$ per la media incognita

  `Variance` varianza campionaria

  `Stdev` deviazione standard

  `Skewness` asimmetria campionaria

  `Kurtosis` kurtosi campionaria

- **Formula:**

  `nobs`
  $$n$$

  `NAs`
  $$\#\texttt{NA} \quad + \quad \#\texttt{NaN}$$

  `Minimum`
  $$x_{(1)}$$

  `Maximum`
  $$x_{(m)}$$

  `1. Quartile`
  $$Q_{0.25}(x)$$

  `3. Quartile`
  $$Q_{0.75}(x)$$

  `Mean`
  $$\bar{x}$$

  `Median`
  $$Q_{0.5}(x)$$

  `Sum`
  $$\sum_{i=1}^{m} x_i$$

  `SE Mean`
  $$s_x \,/\, \sqrt{m}$$

  `LCL Mean`
  $$\bar{x} - t_{1-\alpha\,/\,2,\,m-1}\, s_x \,/\, \sqrt{m}$$

UCL Mean

$$\bar{x} + t_{1-\alpha\,/\,2,\,m-1}\, s_x \,/\, \sqrt{m}$$

Variance

$$s_x^2$$

Stdev

$$s_x$$

Skewness

$$\frac{1}{m}\sum_{i=1}^{m}\left(\frac{x_i-\bar{x}}{s_x}\right)^3$$

Kurtosis

$$\frac{1}{m}\sum_{i=1}^{m}\left(\frac{x_i-\bar{x}}{s_x}\right)^4 - 3$$

- **Examples:**

```
> x <- c(1, 2.3, 5, 6.7, 8)
> 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, ci = 0.95)

            round.ans..digits...6.
nobs                      5.000000
NAs                       0.000000
Minimum                   1.000000
Maximum                   8.000000
1. Quartile               2.300000
3. Quartile               6.700000
Mean                      4.600000
Median                    5.000000
Sum                      23.000000
SE Mean                   1.311106
LCL Mean                  0.959785
UCL Mean                  8.240215
Variance                  8.595000
Stdev                     2.931723
Skewness                 -0.113076
Kurtosis                  1.476555

> x <- c(1.3, NaN, 2, 3.4, 3.4, 5.7, NA, 3.8, 0, 9, 0)
> n <- 11
> m <- 11 - sum(is.na(x))
> m

[1] 9

> sum(is.na(x))

[1] 2

> min(x, na.rm = TRUE)

[1] 0
```

```
> max(x, na.rm = TRUE)

[1] 9

> quantile(x, probs = 0.25, na.rm = TRUE)

25%
1.3

> quantile(x, probs = 0.75, na.rm = TRUE)

75%
3.8

> mean(x, na.rm = TRUE)

[1] 3.177778

> median(x, na.rm = TRUE)

[1] 3.4

> sum(x, na.rm = TRUE)

[1] 28.6

> sd(x, na.rm = TRUE)/sqrt(m)

[1] 0.9563788

> alpha <- 0.05
> mean(x, na.rm = TRUE) - qt(1 - alpha/2, m - 1) * sd(x, na.rm = TRUE)/sqrt(m)

[1] 0.9723642

> mean(x, na.rm = TRUE) + qt(1 - alpha/2, m - 1) * sd(x, na.rm = TRUE)/sqrt(m)

[1] 5.383191

> var(x, na.rm = TRUE)

[1] 8.231944

> sd(x, na.rm = TRUE)

[1] 2.869137

> mean((x - mean(x, na.rm = TRUE))^3/sd(x, na.rm = TRUE)^3, na.rm = TRUE)

[1] 0.6644322

> mean((x - mean(x, na.rm = TRUE))^4/sd(x, na.rm = TRUE)^4, na.rm = TRUE) -
+     3

[1] -0.6913239

> basicStats(x, ci = 0.95)
```

```
             round.ans..digits...6.
nobs                      11.000000
NAs                        2.000000
Minimum                    0.000000
Maximum                    9.000000
1. Quartile                1.300000
3. Quartile                3.800000
Mean                       3.177778
Median                     3.400000
Sum                       28.600000
SE Mean                    0.956379
LCL Mean                   0.972364
UCL Mean                   5.383191
Variance                   8.231944
Stdev                      2.869137
Skewness                   0.792829
Kurtosis                   2.921918
```

- **Note 1:** Calcola le statistiche descrittive utilizzando `x` privato dei valori `NA` e `NaN`.
- **Note 2:** Vale la relazione $m = n -$ (`#NA + #NaN`).
- **Note 3:** Calcola i quartili con la funzione `quantile()`.

stat.desc()

- **Package:** `pastecs`
- **Input:**

  `x` vettore numerico di dimensione n

  `p` livello di confidenza $1 - \alpha$
- **Description:** statistiche descrittive
- **Output:**

  `nbr.val` dimensione campionaria m di `x` privato dei valori `NA` e `NaN`

  `nbr.null` numero di valori nulli

  `nbr.na` numero di valori `NA` e `NaN`

  `min` minimo

  `max` massimo

  `range` campo di variazione

  `sum` somma

  `median` mediana

  `mean` media aritmetica

  `SE.mean` errore standard della media

  `CI.mean.p` ampiezza dell'intervallo di confidenza a livello $1 - \alpha$

  `var` varianza campionaria

  `std.dev` deviazione standard

  `coef.var` coefficiente di variazione campionario
- **Formula:**

  `nbr.val`

  $$m$$

  `nbr.null`

  $$\#\,\texttt{0}$$

  `nbr.na`

  $$\#\,\texttt{NA} \quad + \quad \#\,\texttt{NaN}$$

min

$$x_{(1)}$$

max

$$x_{(m)}$$

range

$$x_{(m)} - x_{(1)}$$

sum

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

median

$$Q_{0.5}(x)$$

mean

$$\bar{x}$$

SE.mean

$$s_x \,/\, \sqrt{m}$$

CI.mean.p

$$t_{1-\alpha\,/\,2,\,m-1}\, s_x \,/\, \sqrt{m}$$

var

$$s_x^2$$

std.dev

$$s_x$$

coef.var

$$s_x \,/\, \bar{x}$$

- **Examples:**

```
> x <- c(1, 2.3, 5, 6.7, 8)
> length(x)

[1] 5

> sum(x == 0)

[1] 0

> sum(is.na(x))

[1] 0

> min(x)

[1] 1

> max(x)

[1] 8

> max(x) - min(x)

[1] 7

> sum(x)

[1] 23
```

```
> median(x)

[1] 5

> mean(x)

[1] 4.6

> sd(x)/sqrt(length(x))

[1] 1.311106

> alpha <- 0.05
> qt(1 - alpha/2, df = length(x) - 1) * sd(x)/sqrt(length(x))

[1] 3.640215

> var(x)

[1] 8.595

> sd(x)

[1] 2.931723

> sd(x)/mean(x)

[1] 0.6373311

> stat.desc(x, p = 0.95)

     nbr.val     nbr.null       nbr.na          min          max        range
   5.0000000    0.0000000    0.0000000    1.0000000    8.0000000    7.0000000
         sum       median         mean      SE.mean CI.mean.0.95          var
  23.0000000    5.0000000    4.6000000    1.3111064    3.6402150    8.5950000
     std.dev     coef.var
   2.9317230    0.6373311

> x <- c(1.3, NaN, 2, 3.4, 3.4, 5.7, NA, 3.8, 0, 9, 0)
> n <- 11
> m <- 11 - sum(is.na(x))
> m

[1] 9

> sum(x == 0, na.rm = TRUE)

[1] 2

> sum(is.na(x))

[1] 2

> min(x, na.rm = TRUE)

[1] 0

> max(x, na.rm = TRUE)
```

```
[1] 9

> max(x, na.rm = TRUE) - min(x, na.rm = TRUE)

[1] 9

> sum(x, na.rm = TRUE)

[1] 28.6

> median(x, na.rm = TRUE)

[1] 3.4

> mean(x, na.rm = TRUE)

[1] 3.177778

> sd(x, na.rm = TRUE)/sqrt(m)

[1] 0.9563788

> alpha <- 0.05
> qt(1 - alpha/2, df = m - 1) * sd(x, na.rm = TRUE)/sqrt(m)

[1] 2.205414

> var(x, na.rm = TRUE)

[1] 8.231944

> sd(x, na.rm = TRUE)

[1] 2.869137

> sd(x, na.rm = TRUE)/mean(x, na.rm = TRUE)

[1] 0.9028751

> stat.desc(x, p = 0.95)

     nbr.val     nbr.null       nbr.na          min          max        range
   9.0000000    2.0000000    2.0000000    0.0000000    9.0000000    9.0000000
         sum       median         mean      SE.mean CI.mean.0.95          var
  28.6000000    3.4000000    3.1777778    0.9563788    2.2054136    8.2319444
     std.dev     coef.var
   2.8691365    0.9028751
```

- **Note 1:** Calcola le statistiche descrittive utilizzando `x` privato dei valori `NA` e `NaN`.
- **Note 2:** Vale la relazione $m = n -$ (`#NA + #NaN`).
- **Note 3:** Calcola i quartili con la funzione `quantile()`.

boxplot.stats()

- **Package:** grDevices
- **Input:**

  `x` vettore numerico di dimensione n

  `coef` valore c positivo
- **Description:** 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`

  $$x_{(1)} \qquad Q_{0.5}\left(x_i\,|_{x_i \leq Q_{0.5}(x)}\right) \qquad Q_{0.5}(x) \qquad Q_{0.5}\left(x_i\,|_{x_i \geq Q_{0.5}(x)}\right) \qquad x_{(n)}$$

  `n`

  $$n$$

  `conf`

  $$Q_{0.5}(x) \mp 1.58 \cdot IQR(x) \, / \, \sqrt{n}$$

  `out`

  $$x_i < Q_{0.25}(x) - c \cdot IQR(x) \quad OR \quad x_i > Q_{0.75}(x) + c \cdot IQR(x)$$
- **Examples:**

```
> x <- c(1.2, 1.2, 2.2, 3, 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 = 1.4)$stats

[1]  1.2  1.2  2.6 15.6 15.6

> n <- 6
> boxplot.stats(x, coef = 1.4)$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 = 1.4)$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 = 1.4)$out
```

```
[1] 71.6

> x <- c(1, 2.3, 5, 6.7, 8)
> c <- 2.6
> fn <- fivenum(x)
> fn

[1] 1.0 2.3 5.0 6.7 8.0

> boxplot.stats(x, coef = 2.6)$stats

[1] 1.0 2.3 5.0 6.7 8.0

> n <- 5
> boxplot.stats(x, coef = 2.6)$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 = 2.6)$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 = 2.6)$out

numeric(0)
```

- **Note:** Calcola i quartili con la funzione `fivenum()`.

3.18 Distribuzione di frequenza

tabulate()

- **Package:** `base`
- **Input:**

  `bin` vettore di valori naturali di dimensione n
- **Description:** distribuzione di frequenza per i valori naturali $1, 2, \ldots, \max(\texttt{bin})$
- **Examples:**

```
> 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
- **Input:**

  x vettore alfanumerico di dimensione n

- **Description:** distribuzione di frequenza
- **Examples:**

```
> x <- c("a", "a", "b", "c", "a", "c")
> table(x)

x
a b c
3 1 2

> table(x)/length(x)

x
        a         b         c
0.5000000 0.1666667 0.3333333

> f <- factor(c("a", "b", "c", "b", "a", "c", "a", "b", "b", "c",
+     "a"))
> f

 [1] a b c b a c a b b c a
Levels: a b c

> g <- factor(c("A", "S", "A", "S", "S", "S", "A", "S", "S", "A",
+     "A"))
> 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 <- c(1, 2, 3, 2, 1, 3, 1, 1, 2, 3)
> table(x)

x
1 2 3
4 3 3
```

unique()

- **Package:** base
- **Input:**

  x vettore alfanumerico di dimensione n

- **Description:** supporto (valori distinti di x)
- **Examples:**

```
> x <- c("a", "a", "b", "c", "a", "c")
> unique(x)

[1] "a" "b" "c"

> x <- c(1, 2, 3, 2, 1, 3, 1, 1, 2, 3)
> unique(x)

[1] 1 2 3

> x <- c(12, -3, 7, 12, 4, -3, 12, 7, -3)
> x[!duplicated(x)]

[1] 12 -3  7  4

> unique(x)

[1] 12 -3  7  4
```

duplicated()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

- **Description:** segnalazione di valori duplicati
- **Examples:**

```
> x <- c(1, 2, 1, 3, 2, 2, 4)
> duplicated(x)

[1] FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE

> x <- c(1, 2, 1, 2, 1, 2)
> duplicated(x)

[1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE

> x <- c(12, -3, 7, 12, 4, -3, 12, 7, -3)
> unique(x[duplicated(x)])

[1] 12 -3  7
```

3.19 Istogramma

hist()

- **Package:** graphics
- **Input:**

  x vettore numerico di dimensione n

  breaks estremi delle classi di ampiezza b_i

  right = TRUE / FALSE classi chiuse a destra $\left(a_{(i)},\, a_{(i+1)}\right]$ oppure a sinistra $\left[a_{(i)},\, a_{(i+1)}\right)$

  include.lowest = TRUE / FALSE estremo incluso

  plot = FALSE
- **Description:** 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

  $$\frac{a_{(i)} + a_{(i+1)}}{2} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, m-1$$
- **Examples:**

```
> x <- c(51.1, 52.3, 66.7, 77.1, 77.15, 77.17)
> n <- 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 = FALSE, include.lowest = FALSE, plot = FALSE)$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 = FALSE, include.lowest = FALSE, plot = FALSE)$counts

[1] 2 1 3

> count/(n * b)

[1] 0.02222222 0.03333333 0.03333333

> hist(x, breaks = a, right = FALSE, include.lowest = FALSE, plot = FALSE)$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 = FALSE, include.lowest = FALSE, plot = FALSE)$mids

[1] 57.5 67.5 77.5

> x <- c(1, 1.2, 2.2, 2.3, 3, 5, 6.7, 8, 15.6)
> n <- 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 = FALSE, include.lowest = FALSE, plot = FALSE)$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 = FALSE, include.lowest = FALSE, plot = FALSE)$counts
```

```
[1] 5 3 0 1

> count/(n * b)

[1] 0.11111111 0.06666667 0.00000000 0.02222222

> hist(x, breaks = a, right = FALSE, include.lowest = FALSE, plot = FALSE)$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 = FALSE, include.lowest = FALSE, plot = FALSE)$mids

[1]  2.5  7.5 12.5 17.5
```

n.bins()

- **Package:** `car`
- **Input:**

  `x` vettore numerico di dimensione n

  `rule = "freedman.diaconis" / "sturges" / "scott" / "simple"` algoritmo

- **Description:** algoritmo di calcolo per il numero di classi di un istogramma
- **Formula:**

`rule = "freedman.diaconis"`

$$n_c = \left\lceil \frac{x_{(n)} - x_{(1)}}{2 \; IQR(x) \; n^{-1/3}} \right\rceil$$

`rule = "sturges"`

$$n_c = \lceil \log_2(n) + 1 \rceil$$

`rule = "scott"`

$$n_c = \left\lceil \frac{x_{(n)} - x_{(1)}}{3.5 \, s_x \, n^{-1/3}} \right\rceil$$

`rule = "simple"`

$$n_c = \begin{cases} \lfloor 2 \sqrt{n} \rfloor & \text{se } n \le 100 \\ \lfloor 10 \log_{10}(n) \rfloor & \text{se } n > 100 \end{cases}$$

- **Examples:**

```
> x <- c(2.3, 1, 5, 6.7, 8)
> x <- sort(x)
> x

[1] 1.0 2.3 5.0 6.7 8.0
```

```
> n <- 5
> nc <- ceiling((x[n] - x[1])/(2 * IQR(x) * n^(-1/3)))
> nc

[1] 2

> n.bins(x, rule = "freedman.diaconis")

[1] 2

> x <- c(2.3, 1, 5, 6.7, 8)
> n <- 5
> nc <- ceiling(log2(n) + 1)
> nc

[1] 4

> n.bins(x, rule = "sturges")

[1] 4

> x <- c(2.3, 1, 5, 6.7, 8)
> x <- sort(x)
> x

[1] 1.0 2.3 5.0 6.7 8.0

> n <- 5
> sx <- sd(x)
> nc <- ceiling((x[n] - x[1])/(3.5 * sx * n^(-1/3)))
> nc

[1] 2

> n.bins(x, rule = "scott")

[1] 2

> x <- c(2.3, 1, 5, 6.7, 8)
> n <- 5
> nc <- floor(2 * sqrt(n))
> nc

[1] 4

> n.bins(x, rule = "simple")

[1] 4
```

- **Note:** Calcola i quartili con la funzione `quantile()`.

nclass.FD()

- **Package:** grDevices
- **Input:**

  x vettore numerico di dimensione n

- **Description:** numero di classi di un istogramma secondo *Freedman - Diaconis*
- **Formula:**

$$n_c = \left\lceil \frac{x_{(n)} - x_{(1)}}{2\; IQR(x)\; n^{-1/3}} \right\rceil$$

- **Examples:**

```
> x <- c(2.3, 1, 5, 6.7, 8)
> x <- sort(x)
> x

[1] 1.0 2.3 5.0 6.7 8.0

> n <- 5
> nc <- ceiling((x[n] - x[1])/(2 * IQR(x) * n^(-1/3)))
> nc

[1] 2

> nclass.FD(x)

[1] 2

> x <- c(3.4, 5.52, 6.4, 7.56, 8.7, 8.6, 5.4, 5.5)
> x <- sort(x)
> x <- c(3.4, 5.4, 5.5, 5.52, 6.4, 7.56, 8.6, 8.7)
> n <- 8
> nc <- ceiling((x[n] - x[1])/(2 * IQR(x) * n^(-1/3)))
> nc

[1] 3

> nclass.FD(x)

[1] 3
```

- **Note:** Calcola i quartili con la funzione quantile().

nclass.Sturges()

- **Package:** grDevices
- **Input:**

  x vettore numerico di dimensione n

- **Description:** numero di classi di un istogramma secondo *Sturges*
- **Formula:**

$$n_c = \lceil \log_2(n) + 1 \rceil$$

- **Examples:**

```
> x <- c(1, 2.3, 5, 6.7, 8)
> n <- 5
> nc <- ceiling(log2(n) + 1)
> nc

[1] 4

> nclass.Sturges(x)

[1] 4

> x <- c(3.4, 5.4, 5.5, 5.52, 6.4, 7.56, 8.6, 8.7)
> n <- 8
> nc <- ceiling(log2(n) + 1)
> nc

[1] 4

> nclass.Sturges(x)

[1] 4
```

nclass.scott()

- **Package:** grDevices
- **Input:**

  x vettore numerico di dimensione n
- **Description:** numero di classi di un istogramma secondo *Scott*
- **Formula:**

$$n_c = \left\lceil \frac{x_{(n)} - x_{(1)}}{3.5 \, s_x \, n^{-1\,/\,3}} \right\rceil$$

- **Examples:**

```
> x <- c(2.3, 1, 5, 6.7, 8)
> x <- sort(x)
> x

[1] 1.0 2.3 5.0 6.7 8.0

> n <- 5
> sx <- sd(x)
> nc <- ceiling((x[n] - x[1])/(3.5 * sx * n^(-1/3)))
> nc

[1] 2

> nclass.scott(x)

[1] 2

> x <- c(3.4, 5.4, 5.5, 5.52, 6.4, 7.56, 8.6, 8.7)
> x <- sort(x)
> x

[1] 3.40 5.40 5.50 5.52 6.40 7.56 8.60 8.70
```

```
> n <- 8
> sx <- sd(x)
> nc <- ceiling((x[n] - x[1])/(3.5 * sx * n^(-1/3)))
> nc

[1] 2

> nclass.scott(x)

[1] 2
```

3.20 Variabili casuali 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

$p_X(x) = \binom{m}{x} p^x (1-p)^{m-x} \quad x = 0, 1, 2, \ldots, m, \quad m \in \mathbb{N} / \{0\}, \quad 0 < p < 1$

$\mu_X = m\,p$

$\sigma_X^2 = m\,p\,(1-p)$

Binomiale Negativa

$p_X(x) = \binom{r+x-1}{x} p^r (1-p)^x = \binom{r+x-1}{r-1} p^r (1-p)^x \quad x \in \mathbb{N}, \quad r \in \mathbb{N}\backslash\{0\}, \quad 0 < p < 1$

$\mu_X = r\,(1-p) / p$

$\sigma_X^2 = r\,(1-p) / p^2$

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

$p_X(x) = p\,(1-p)^{x-1} \quad x \in \mathbb{N}\backslash\{0\}, \quad 0 < p < 1$

$\mu_X = 1 / p$

$\sigma_X^2 = (1-p) / p^2$

Ipergeometrica

$p_X(x) = \binom{M}{x} \binom{N-M}{k-x} / \binom{N}{k}$

$x = 0, 1, 2, \ldots, k$

$N \in \mathbb{N}\backslash\{0\}$

$k = 1, 2, \ldots, N$

$M = 0, 1, 2, \ldots, N-1$

$\mu_X = k\,(M\,/\,N)$

$\sigma^2_X = k\,(M\,/\,N)\,(1 - M\,/\,N)\,(N-k)\,/\,(N-1)$

Multinomiale

$p_{X_1,\,X_2,\,\ldots,\,X_k}(x_1,\,x_2,\,\ldots,\,x_k) = \frac{m\,!}{x_1\,!\,x_2\,!\cdots x_k\,!}\prod_{i=1}^{k} p_i^{x_i}$

$x_i = 0, 1, 2, \ldots, m \quad \forall i = 1, 2, \ldots, k$

$0 < p_i < 1 \quad \forall i = 1, 2, \ldots, k$

$\sum_{i=1}^{k} x_i = m$

$\sum_{i=1}^{k} p_i = 1$

$\mu_{X_i} = m\,p_i \quad \forall i = 1, 2, \ldots, k$

$\sigma^2_{X_i} = m\,p_i\,(1-p_i) \quad \forall i = 1, 2, \ldots, k$

$\sigma_{X_i X_j} = -m\,p_i\,p_j \quad \forall i \neq j = 1, 2, \ldots, k$

Poisson

$p_X(x) = \lambda^x\,e^{-\lambda}\,/\,x\,! \quad x \in \mathbb{N}, \quad \lambda > 0$

$\mu_X = \lambda$

$\sigma^2_X = \lambda$

Tavola argomenti comandi R

Variabile Casuale	Suffisso	Parametri	Package
Bernoulli	`binom`	`size, prob`	`stats`
Binomiale	`binom`	`size, prob`	`stats`
Binomiale Negativa	`nbinom`	`size, prob`	`stats`
Geometrica	`geom`	`prob`	`stats`
Geometrica 2	`geomet`	`p`	`distributions`
Ipergeometrica	`hyper`	`m, n, k`	`stats`
Multinomiale	`multinom`	`size, prob`	`stats`
Poisson	`pois`	`lambda`	`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)
	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)
	Random	rbinom(n,size=m,prob=p)
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)
	Random	rnbinom(n,size=r,prob=p)
Geometrica	**Densità**	dgeom(x=x,prob=p)
	Ripartizione	pgeom(q=x,prob=p)
	Quantile	qgeom(p=α,prob=p)
	Random	rgeom(n,prob=p)

Geometrica 2	**Densità**	geometpdf(p=p,x=x)
	Ripartizione	geometcdf(p=p,x=x)
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)
	Random	rhyper(nn,m=M,n=$N-M$,k=k)
Multinomiale	**Densità**	dmultinom(x=c($x_1,\ldots,x_k$),prob=c($p_1,\ldots,p_k$))
	Random	rmultinom(n,size=m,prob=c($p_1,\ldots,p_k$))
Poisson	**Densità**	dpois(x=x,lambda=λ)
	Ripartizione	ppois(q=x,lambda=λ)
	Quantile	qpois(p=α,lambda=λ)
	Random	rpois(n,lambda=λ)

3.21 Variabili casuali continue

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 NC

$\frac{\chi^2_\theta(\delta)}{\chi^2_\theta(\delta)+\chi^2_\lambda} \quad 0<x<1, \quad \theta>0, \quad \lambda>0, \quad \delta>0$

Burr

$f_X(x) \;=\; \frac{\theta\,\mu\,(x\,/\,\lambda)^\theta}{x\left(1+(x\,/\,\lambda)^\theta\right)^{\mu+1}} \quad x>0, \quad \theta>0, \quad \mu>0, \quad \lambda>0$

$\mu_X \;=\; \lambda\,\Gamma(1-1\,/\,\theta)\,\Gamma(1\,/\,\theta+\mu)\,/\,\Gamma(\mu)$

$\sigma_X^2 \;=\; \left[\Gamma(\mu)\,\Gamma(1-2\,/\,\theta)\,\Gamma(2\,/\,\theta+\mu)-\Gamma^2(1-1\,/\,\theta)\,\Gamma(1\,/\,\theta+\mu)\right]\,\lambda^2\,/\,\Gamma^2(\mu) \quad \text{per } \theta>2$

Cauchy

$f_X(x) \;=\; (\pi\,\lambda)^{-1}\,\left[1+((x-\theta)\,/\,\lambda)^2\right]^{-1} \quad x\in\mathbb{R}, \quad \theta\in\mathbb{R}, \quad \lambda>0$

$\mu_X \;=\; \nexists$

$\sigma_X^2 \;=\; \nexists$

Chi - Quadrato

$f_X(x) \;=\; \frac{2^{-k\,/\,2}}{\Gamma(k\,/\,2)}\,x^{(k-2)\,/\,2}\,e^{-x\,/\,2} \quad x>0, \quad k>0$

$\mu_X \;=\; k$

$\sigma_X^2 \;=\; 2\,k$

Chi - Quadrato NC

$f_X(x) \;=\; \exp\left(-(x+\delta)\,/\,2\right)\,\sum_{i=0}^{\infty}\frac{(\delta\,/\,2)^i\,x^{k\,/\,2+i-1}}{2^{k\,/\,2+i}\,\Gamma(k\,/\,2+i)\,i\,!} \quad x>0, \quad k>0, \quad \delta>0$

$\mu_X \;=\; k+\delta$

$\sigma_X^2 \;=\; 2\,(k+2\,\delta)$

Dirichlet

$f_{X_1, X_2, \dots, X_k}(x_1, x_2, \dots, x_k) = \frac{\Gamma(\alpha_1+\alpha_2+\cdots+\alpha_k)}{\Gamma(\alpha_1)\,\Gamma(\alpha_2)\cdots\Gamma(\alpha_k)} \prod_{i=1}^{k} x_i^{\alpha_i - 1}$

$x_i > 0 \quad \forall i = 1, 2, \dots, k$

$\alpha_i > 0 \quad \forall i = 1, 2, \dots, k$

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

$\sum_{i=1}^{k} \alpha_i = \alpha$

$\mu_{X_i} = \frac{\alpha_i}{\alpha} \quad \forall i = 1, 2, \dots, k$

$\sigma^2_{X_i} = \frac{\alpha_i\,(\alpha-\alpha_i)}{\alpha^2\,(\alpha+1)} \quad \forall i = 1, 2, \dots, k$

$\sigma_{X_i X_j} = -\frac{\alpha_i\,\alpha_j}{\alpha^2\,(\alpha+1)} \quad \forall i \neq j = 1, 2, \dots, k$

Esponenziale

$f_X(x) = \lambda\, e^{-\lambda\, x} \quad x > 0, \quad \lambda > 0$

$\mu_X = 1\,/\,\lambda$

$\sigma^2_X = 1\,/\,\lambda^2$

Fisher

$f_X(x) = \frac{\Gamma((n_1+n_2)\,/\,2)}{\Gamma(n_1\,/\,2)\,\Gamma(n_2\,/\,2)} \left(\frac{n_1}{n_2}\right)^{n_1\,/\,2} x^{(n_1-2)\,/\,2} \left(1 + \frac{n_1}{n_2}\,x\right)^{-(n_1+n_2)\,/\,2} \quad x,\, n_1,\, n_2 > 0$

$\mu_X = \frac{n_2}{n_2-2}$ per $n_2 > 2$

$\sigma^2_X = \frac{2\,n_2^2\,(n_1+n_2-2)}{n_1\,(n_2-2)^2\,(n_2-4)}$ per $n_2 > 4$

Fisher NC

$f_X(x) = \frac{n_1^{n_1\,/\,2}\,n_2^{n_2\,/\,2}}{\exp\,(\delta\,/\,2)} \frac{x^{n_1\,/\,2-1}}{(n_1\,x+n_2)^{(n_1+n_2)\,/\,2}} \sum_{i=0}^{\infty} \frac{(\delta\,/\,2)^i}{i\,!} \frac{\Gamma(n_1\,/\,2+n_2\,/\,2+i)}{\Gamma(n_1\,/\,2+i)\,\Gamma(n_2\,/\,2)} \left(\frac{n_1\,x}{n_1\,x+n_2}\right)^i \quad x,\, n_1,\, n_2,\, \delta > 0$

$\mu_X = \frac{n_2\,(n_1+\delta)}{n_1\,(n_2-2)}$ per $n_2 > 2$

$\sigma^2_X = 2\left(\frac{n_2}{n_1}\right)^2 \frac{(n_1+\delta)^2+(n_1+2\,\delta)\,(n_2-2)}{(n_2-2)^2\,(n_2-4)}$ per $n_2 > 4$

Friedman

$x > 0 \quad r \in \mathbb{N}\,/\,\{0,1\}, \quad N \in \mathbb{N}\,/\,\{0,1\}$

Gamma

$f_X(x) = \frac{\lambda^\theta}{\Gamma(\theta)}\, x^{\theta-1}\, e^{-\lambda\, x} \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X = \theta\,/\,\lambda$

$\sigma^2_X = \theta\,/\,\lambda^2$

Gamma 2

$f_X(x) = \frac{1}{\lambda^\theta\,\Gamma(\theta)}\, x^{\theta-1}\, e^{-x\,/\,\lambda} \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X = \theta\,\lambda$

$\sigma^2_X = \theta\,\lambda^2$

Gamma inversa

$f_X(x) \,=\, \frac{\lambda^\theta}{\Gamma(\theta)}\, x^{-\,(\theta+1)}\, e^{-\lambda\,/\,x} \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X \,=\, \lambda\,/\,(\theta - 1)$ per $\theta > 1$

$\sigma_X^2 \,=\, \lambda^2\,/\,[(\theta-1)^2\,(\theta-2)]$ per $\theta > 2$

Gamma inversa 2

$f_X(x) \,=\, \frac{1}{\lambda^\theta\,\Gamma(\theta)}\, x^{-\,(\theta+1)}\, e^{-1\,/\,(\lambda\,x)} \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X \,=\, 1\,/\,[\lambda\,(\theta-1)]$ per $\theta > 1$

$\sigma_X^2 \,=\, 1\,/\,[\lambda^2\,(\theta-1)^2\,(\theta-2)]$ per $\theta > 2$

Laplace

$f_X(x) \,=\, \frac{1}{2}\,\lambda^{-1}\,\exp\left(-\frac{|\,x-\theta\,|}{\lambda}\right) \quad x \in \mathbb{R}, \quad \theta \in \mathbb{R}, \quad \lambda > 0$

$\mu_X \,=\, \theta$

$\sigma_X^2 \,=\, 2\,\lambda^2$

Logistica

$f_X(x) \,=\, \lambda^{-1}\,\exp\left((x-\theta)\,/\,\lambda\right)\,\left(1+\exp\left((x-\theta)\,/\,\lambda\right)\right)^{-2} \quad x \in \mathbb{R}, \quad \theta \in \mathbb{R}, \quad \lambda > 0$

$\mu_X \,=\, \theta$

$\sigma_X^2 \,=\, (\pi\,\lambda)^2\,/\,3$

LogLogistica

$f_X(x) \,=\, \frac{\theta\,(x\,/\,\lambda)^\theta}{x\left(1+(x\,/\,\lambda)^\theta\right)^2} \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X \,=\, \lambda\,\Gamma(1-1\,/\,\theta)\,\Gamma(1\,/\,\theta+1)$

$\sigma_X^2 \,=\, \left[\Gamma(1-2\,/\,\theta)\,\Gamma(2\,/\,\theta+1) - \Gamma^2(1-1\,/\,\theta)\,\Gamma(1\,/\,\theta+1)\right]\,\lambda^2$ per $\theta > 2$

LogNormale

$f_X(x) \,=\, \left(\sigma\,x\,\sqrt{2\,\pi}\right)^{-1}\,\exp\left(-(\log(x)-\mu)^2\,/\,(2\,\sigma^2)\right) \quad x > 0, \quad \mu \in \mathbb{R}, \sigma > 0$

$\mu_X \,=\, \exp\left(\mu + \sigma^2\,/\,2\right)$

$\sigma_X^2 \,=\, \exp\left(2\,\mu+\sigma^2\right)\,\left(\exp\left(\sigma^2\right)-1\right)$

Mann - Whitney

$0 \leq x \leq n_x\,n_y, \quad n_x \in \mathbb{N}\,/\,\{0\}, \quad n_y \in \mathbb{N}\,/\,\{0\}$

$\mu_X \,=\, n_x\,n_y\,/\,2$

$\sigma_X^2 \,=\, n_x\,n_y\,(n_x+n_y+1)\,/\,12$

Normale

$f_X(x) \,=\, \left(2\,\pi\,\sigma^2\right)^{-1\,/\,2}\,\exp\left(-(x-\mu)^2\,/\,(2\,\sigma^2)\right) \quad x \in \mathbb{R}, \quad \mu \in \mathbb{R}, \quad \sigma > 0$

$\mu_X \,=\, \mu$

$\sigma_X^2 \,=\, \sigma^2$

Normale doppia

$$f_{X_1,X_2}(x_1,x_2) = \frac{1}{2\,\pi\,\sqrt{\sigma_{11}\,\sigma_{22}\,(1-\rho^2)}} \exp\left(-\frac{1}{2\,(1-\rho^2)}\left[\left(\frac{x_1-\mu_1}{\sqrt{\sigma_{11}}}\right)^2 - 2\,\rho\,\frac{x_1-\mu_1}{\sqrt{\sigma_{11}}}\,\frac{x_2-\mu_2}{\sqrt{\sigma_{22}}} + \left(\frac{x_2-\mu_2}{\sqrt{\sigma_{22}}}\right)^2\right]\right)$$

$x_i \in \mathbb{R} \quad \forall i = 1, 2$

$\mu_i \in \mathbb{R} \quad \forall i = 1, 2$

$\rho = \sigma_{12} / \sqrt{\sigma_{11}\,\sigma_{22}} = \sigma_{21} / \sqrt{\sigma_{11}\,\sigma_{22}} \in (0, 1)$

$V_2 = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix}$ definita positiva

$\sigma_{ii} > 0 \quad \forall i = 1, 2$

$\mu_{X_i} = \mu_i \quad \forall i = 1, 2$

$\sigma^2_{X_i} = \sigma_{ii} \quad \forall i = 1, 2$

$\sigma_{X_1 X_2} = \sigma_{12} = \sigma_{21}$

Normale multipla

$$f_{X_1,X_2,\ldots,X_k}(x_1,x_2,\ldots,x_k) = \frac{1}{(2\,\pi)^{k\,/\,2}\sqrt{\det(V_k)}} \exp\left(-\frac{1}{2}\,(x_1-\mu_1,x_2-\mu_2,\ldots,x_k-\mu_k)^T\,V_k^{-1}\,(x_1-\mu_1,x_2-\mu_2,\ldots,x_k-\mu_k)\right)$$

$x_i \in \mathbb{R} \quad \forall i = 1, 2, \ldots, k$

$\mu_i \in \mathbb{R} \quad \forall i = 1, 2, \ldots, k$

$V_k = \begin{pmatrix} \sigma_{11} & \sigma_{12} & \ldots & \sigma_{1k} \\ \sigma_{21} & \sigma_{22} & \ldots & \sigma_{2k} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{k1} & \sigma_{k2} & \ldots & \sigma_{kk} \end{pmatrix}$ definita positiva

$\sigma_{ii} > 0 \quad \forall i = 1, 2, \ldots, k$

$\mu_{X_i} = \mu_i \quad \forall i = 1, 2, \ldots, k$

$\sigma^2_{X_i} = \sigma_{ii} \quad \forall i = 1, 2, \ldots, k$

$\sigma_{X_i X_j} = \sigma_{ij} = \sigma_{ji} \quad \forall i \neq j = 1, 2, \ldots, k$

Pareto

$f_X(x) = \frac{\theta\,\lambda^\theta}{x^{\theta+1}} \quad x > \lambda, \quad \theta > 0, \quad \lambda > 0$

$\mu_X = \theta\,\lambda\,/\,(\lambda - 1)$

$\sigma^2_X = \theta\,\lambda^2\,/\,\left((\theta-2)\,(\theta-1)^2\right)$ per $\theta > 2$

Student

$f_X(x) = \frac{\Gamma((k+1)\,/\,2)}{\Gamma(k\,/\,2)}\,(k\,\pi)^{-1\,/\,2}\,(1 + x^2\,/\,k)^{-(k+1)\,/\,2} \quad x \in \mathbb{R}, \quad k > 0$

$\mu_X = 0$ per $k > 1$

$\sigma^2_X = k\,/\,(k-2)$ per $k > 2$

Student NC

$f_X(x) = \frac{k^{k\,/\,2}\,\exp\,(-\delta^2\,/\,2)}{\sqrt{\pi}\,\Gamma(n\,/\,2)\,(k+x^2)^{(k+1)\,/\,2}} \sum_{i=0}^{\infty} \frac{\Gamma((k+i+1)\,/\,2)\,\delta^i}{i\,!} \left(\frac{2\,x^2}{k+x^2}\right)^{i\,/\,2} \quad x \in \mathbb{R}, \quad k > 0, \quad \delta \in \mathbb{R}$

$\mu_X = \sqrt{k\,/\,2}\,\delta\,\Gamma\,((k-1)\,/\,2)\,/\,\Gamma\,(k\,/\,2)$ per $k > 1$

$\sigma^2_X = k\,(1+\delta^2)\,/\,(k-2) - \delta\,(k\,/\,2)\,\left(\Gamma\,((k-1)\,/\,2)\,/\,\Gamma\,(k\,/\,2)\right)^2$ per $k > 2$

Tukey

$x > 0, \quad n \in \mathbb{N} \,/\, \{0, 1, 2\}, \quad p \in \mathbb{N} \,/\, \{0, 1\}$

Uniforme

$f_X(x) \,=\, 1 \,/\, (b - a) \quad a < x < b, \quad a \in \mathbb{R}, \quad b \in \mathbb{R}, \quad a < b$

$\mu_X \,=\, (a + b) \,/\, 2$

$\sigma^2_X \,=\, (b - a)^2 \,/\, 12$

Wald

$f_X(x) \,=\, (\lambda \,/\, (2\,\pi\,x^3))^{1\,/\,2} \,\exp\left(-\lambda\,(x - \theta)^2 \,/\, (2\,\theta^2\,x)\right) \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X \,=\, \theta$

$\sigma^2_X \,=\, \theta^3 \,/\, \lambda$

Weibull

$f_X(x) \,=\, (\theta \,/\, \lambda)\,(x \,/\, \lambda)^{\theta - 1} \,\exp\left(-\,(x \,/\, \lambda)^{\theta}\right) \quad x > 0, \quad \theta > 0, \quad \lambda > 0$

$\mu_X \,=\, \lambda\,\Gamma((\theta + 1) \,/\, \theta)$

$\sigma^2_X \,=\, \lambda^2 \,\left[\Gamma((\theta + 2) \,/\, \theta) - \Gamma^2((\theta + 1) \,/\, \theta)\right]$

Wilcoxon signed rank

$0 \leq x \leq n\,(n + 1) \,/\, 2, \quad n \in \mathbb{N} \,/\, \{0\}$

$\mu_X \,=\, n\,(n + 1) \,/\, 4$

$\sigma^2_X \,=\, n\,(n + 1)\,(2\,n + 1) \,/\, 24$

Tavola argomenti comandi R

Variabile Casuale	**Suffisso**	**Parametri**	**Package**
Beta	beta	shape1, shape2	stats
Beta NC	beta	shape1, shape2, ncp	stats
Burr	burr	shape1, shape2, scale, rate	actuar
Cauchy	cauchy	location, scale	stats
Chi - Quadrato	chisq	df	stats
Chi - Quadrato NC	chisq	df, ncp	stats
Dirichlet	dirichlet	alpha	MCMCpack
Esponenziale	exp	rate	stats
Fisher	f	df1, df2	stats
Fisher NC	f	df1, df2, ncp	stats
Friedman	Friedman	r, N	SuppDists
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
Laplace	laplace	m, s	formularioR
Logistica	logis	location, scale	stats
LogLogistica	llogis	shape, scale, rate	actuar
LogNormale	lnorm	meanlog, sdlog	stats
Mann - Whitney	wilcox	m, n	stats
Normale	norm	mean, sd	stats
Normale doppia	mvnorm	mean, sigma	mvtnorm
Normale multipla	mvnorm	mean, sigma	mvtnorm
Pareto	pareto1	shape, min	actuar

Student	t	df	stats
Student NC	t	df, ncp	stats
Tukey	tukey	nmeans, df	stats
Uniforme	unif	min, max	stats
Wald	invGauss	nu, lambda	SuppDists
Weibull	weibull	shape, scale	stats
Wilcoxon signed rank	signrank	n	stats

Tavola esempi comandi R

Variabile Casuale	Oggetto	Comando in R
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)
	Random	rbeta(n,shape1=θ,shape2=λ)
Beta NC	**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)
	Random	rbeta(n,shape1=θ,shape2=λ,ncp=δ)
Burr	**Densità**	dburr(x=x,shape1=μ,shape2=θ,scale=λ)
		dburr(x=x,shape1=μ,shape2=θ,rate=$1/\lambda$)
	Ripartizione	pburr(q=x,shape1=μ,shape2=θ,scale=λ)
		pburr(q=x,shape1=μ,shape2=θ,rate=$1/\lambda$)
	Quantile	qburr(p=α,shape1=μ,shape2=θ,scale=λ)
		qburr(p=α,shape1=μ,shape2=θ,rate=$1/\lambda$)
	Random	rburr(n,shape1=μ,shape2=θ,scale=λ)
		rburr(n,shape1=μ,shape2=θ,rate=$1/\lambda$)
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)
	Random	rcauchy(n,location=θ,scale=λ)
Chi - Quadrato	**Densità**	dchisq(x=x,df=k)
	Ripartizione	pchisq(q=x,df=k)
	Quantile	qchisq(p=α,df=k)
	Random	rchisq(n,df=k)
Chi - Quadrato NC	**Densità**	dchisq(x=x,df=k,ncp=δ)
	Ripartizione	pchisq(q=x,df=k,ncp=δ)
	Quantile	qchisq(p=α,df=k,ncp=δ)
	Random	rchisq(n,df=k,ncp=δ)
Dirichlet	**Densità**	ddirichlet(x=c($x_1,\ldots,x_k$),alpha=c($\alpha_1,\ldots,\alpha_k$))
	Random	rdirichlet(n,alpha=c($\alpha_1,\ldots,\alpha_k$))
Esponenziale	**Densità**	dexp(x=x,rate=λ)
	Ripartizione	pexp(q=x,rate=λ)
	Quantile	qexp(p=α,rate=λ)
	Random	rexp(n,rate=λ)
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)
	Random	rf(n,df1=n_1,df2=n_2)
Fisher NC	**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)
	Random	rf(n,df1=n_1,df2=n_2,ncp=δ)
Friedman	**Densità**	dFriedman(x=x,r=r,N=N)
	Ripartizione	pFriedman(q=x,r=r,N=N)
	Quantile	qFriedman(p=α,r=r,N=N)
	Random	rFriedman(n,r=r,N=N)
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=$1/\lambda$)
	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)
	Random	rgamma(n, shape=θ, rate=$1/\lambda$) rgamma(n, shape=θ, scale=λ)
Gamma inversa	**Densità**	dinvgamma(x=x, shape=θ, scale=$1/\lambda$)
	Random	rinvgamma(n, shape=θ, scale=λ)
Gamma inversa 2	**Densità**	dinvgamma(x=x, shape=θ, scale=λ)
	Random	rinvgamma(n, shape=θ, scale=$1/\lambda$)
Laplace	**Densità**	dlaplace(x=x, m=θ, s=λ)
	Ripartizione	plaplace(q=x, m=θ, s=λ)
	Quantile	qlaplace(p=α, m=θ, s=λ)
	Random	rlaplace(n, m=θ, s=λ)
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)
	Random	rlogis(n, location=θ, scale=λ)
LogLogistica	**Densità**	dllogis(x=x, shape=θ, scale=λ) dllogis(x=x, shape=θ, rate=$1/\lambda$)
	Ripartizione	pllogis(q=x, shape=θ, scale=λ) pllogis(q=x, shape=θ, rate=$1/\lambda$)
	Quantile	qllogis(p=α, shape=θ, scale=λ) qllogis(p=α, shape=θ, rate=$1/\lambda$)
	Random	rllogis(n, shape=θ, scale=λ) rllogis(n, shape=θ, rate=$1/\lambda$)
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)
	Random	rlnorm(n, meanlog=μ, sdlog=σ)
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)
	Random	rwilcox(nn, m=n_x, n=n_y)
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)
	Random	rnorm(n, mean=μ, sd=σ)
Normale doppia	**Densità**	dmvnorm(x=c(x_1, x_2), mean=c(μ_1, μ_2), sigma=V_2)
	Ripartizione	pmvnorm(u=c(x_1, x_2), mean=c(μ_1, μ_2), sigma=V_2)
	Random	rmvnorm(n, mean=c(μ_1, μ_2), sigma=V_2)
Normale multipla	**Densità**	dmvnorm(x=c($x_1, x_2, \ldots, x_k$), mean=c($\mu_1, \mu_2, \ldots, \mu_k$), sigma=$V_k$)
	Ripartizione	pmvnorm(u=c($x_1, x_2, \ldots, x_k$), mean=c($\mu_1, \mu_2, \ldots, \mu_k$), sigma=$V_k$)
	Random	rmvnorm(n, mean=c($\mu_1, \mu_2, \ldots, \mu_k$), sigma=$V_k$)
Pareto	**Densità**	dpareto1(x=x, shape=θ, min=λ)
	Ripartizione	ppareto1(q=x, shape=θ, min=λ)
	Quantile	qpareto1(p=α, shape=θ, min=λ)
	Random	rpareto1(n, shape=θ, min=λ)
Student	**Densità**	dt(x=x, df=k)
	Ripartizione	pt(q=x, df=k)
	Quantile	qt(p=α, df=k)
	Random	rt(n, df=k)
Student NC	**Densità**	dt(x=x, df=k, ncp=δ)
	Ripartizione	pt(q=x, df=k, ncp=δ)
	Quantile	qt(p=α, df=k, ncp=δ)
	Random	rt(n, df=k, ncp=δ)
Tukey	**Ripartizione**	ptukey(q=x, nmeans=p, df=n)
	Quantile	qtukey(p=α, nmeans=p, df=n)

Uniforme	**Densità** **Ripartizione** **Quantile** **Random**	dunif(x=x,min=a,max=b) punif(q=x,min=a,max=b) qunif(p=α,min=a,max=b) runif(n,min=a,max=b)
Wald	**Densità** **Ripartizione** **Quantile** **Random**	dinvGauss(x=x,nu=θ,lambda=λ) pinvGauss(q=x,nu=θ,lambda=λ) qinvGauss(p=α,nu=θ,lambda=λ) rinvGauss(n,nu=θ,lambda=λ)
Weibull	**Densità** **Ripartizione** **Quantile** **Random**	dweibull(x=x,shape=θ,scale=λ) pweibull(q=x,shape=θ,scale=λ) qweibull(p=α,shape=θ,scale=λ) rweibull(n,shape=θ,scale=λ)
Wilcoxon signed rank	**Densità** **Ripartizione** **Quantile** **Random**	dsignrank(x=x,n=n) psignrank(q=x,n=n) qsignrank(p=α,n=n) rsignrank(nn,n=n)

3.22 Logit

logit()

- **Package:** faraway
- **Input:**

  x vettore numerico di probabilità di dimensione n

- **Description:** trasformazione logit
- **Formula:**

$$\log\left(\frac{x_i}{1 - x_i}\right) \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Examples:**

```
> x <- c(0.2, 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 <- c(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
- **Input:**

  x vettore numerico di dimensione n
- **Description:** trasformazione logit inversa
- **Formula:**

$$\frac{e^{x_i}}{1 + e^{x_i}} = \frac{1}{1 + e^{-x_i}} \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(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 <- c(2.3, 4.5, 6.7, 7.8, 12)
> 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
- **Input:**

  x vettore numerico di dimensione n
- **Description:** trasformazione logit inversa
- **Formula:**

$$\frac{e^{x_i}}{1 + e^{x_i}} = \frac{1}{1 + e^{-x_i}} \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(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 <- c(2.3, 4.5, 6.7, 7.8, 12)
> 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
```

3.23 Serie storiche

length()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n
- **Description:** dimensione campionaria
- **Formula:**

$$n$$

- **Examples:**

```
> x <- c(1.2, 2.3, 4.5, 6.5)
> length(x)

[1] 4

> x <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4)
> length(x)

[1] 7
```

diff()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

  lag il valore d del ritardo

  differences il valore k dell'ordine delle differenze
- **Description:** differenze in una serie storica
- **Formula:**

$$\left(1 - B^d\right)^k x_t \quad \forall\, t = d\,k+1,\, d\,k+2,\, \dots,\, n$$

$$\text{dove} \quad \left(1 - B^d\right)^k = \sum_{j=0}^{k} \binom{k}{j} (-1)^j B^{jd} \qquad B^h x_t = x_{t-h}$$

- **Examples:**

```
> x <- c(1, 2, 4, 3, 5, 6, -9)
> n <- 7
> d <- 2
> k <- 1
> x[(d + 1):n] - x[1:(n - d)]

[1]   3   1   1   3 -14

> diff(x, lag = 2, differences = 1)

[1]   3   1   1   3 -14
```

```
> x <- c(1, 2, 4, 3, 5, 6, -9)
> n <- 7
> d <- 2
> k <- 2
> x[(k * d + 1):n] - 2 * x[(k * d + 1 - d):(n - d)] + x[(k * d +
+     1 - k * d):(n - k * d)]

[1]  -2   2 -15

> diff(x, lag = 2, differences = 2)

[1]  -2   2 -15

> x <- c(2, 6, 10, 9, 9, 8, 9, 9, 10, 12)
> n <- 10
> d <- 2
> k <- 3
> x[(k * d + 1):n] - 3 * x[(k * d + 1 - d):(n - d)] + 3 * x[(k *
+     d + 1 - 2 * d):(n - 2 * d)] - x[(k * d + 1 - k * d):(n -
+     k * d)]

[1] 10  6  0  0

> diff(x, lag = 2, differences = 3)

[1] 10  6  0  0
```

diffinv()

- **Package:** stats
- **Input:**

  `x` vettore numerico di dimensione n

  `lag` il valore d del ritardo

  `differences` il valore k dell'ordine delle differenze

  `xi` valore necessari a ricostruire la serie storica di partenza

- **Description:** operazione inversa del comando `diff()`
- **Examples:**

```
> x <- c(1, 2, 4, 3, 5, 6, -9)
> n <- 7
> d <- 2
> k <- 1
> diff(x, lag = 2, differences = 1)

[1]   3   1   1   3 -14

> diffinv(diff(x, lag = 2, differences = 1), lag = 2, differences = 1,
+     xi = c(1, 2))

[1]  1  2  4  3  5  6 -9

> x <- c(1, 2, 4, 3, 5, 6, -9)
> n <- 7
> d <- 2
> k <- 2
> diff(x, lag = 2, differences = 2)
```

```
[1]  -2   2 -15

> diffinv(diff(x, lag = 2, differences = 2), lag = 2, differences = 2,
+     xi = c(1, 2, 4, 3))

[1]  1  2  4  3  5  6 -9

> x <- c(2, 6, 10, 9, 9, 8, 9, 9, 10, 12)
> n <- 10
> d <- 2
> k <- 3
> diff(x, lag = 2, differences = 3)

[1] 10  6  0  0

> diffinv(diff(x, lag = 2, differences = 3), lag = 2, differences = 3,
+     xi = c(2, 6, 10, 9, 9, 8))

 [1]  2  6 10  9  9  8  9  9 10 12
```

acf()

- **Package:** stats
- **Input:**

  x vettore numerico di dimensione n

  lag.max il valore d del ritardo

  type = "correlation" / "covariance" / "partial" tipo di legame

  demean = TRUE / FALSE centratura

  plot = FALSE
- **Description:** autocovarianza oppure autocorrelazione
- **Output:**

  acf autocovarianza oppure autocorrelazione

  n.used dimensione campionaria

  lag il valore d del ritardo
- **Formula:**

  acf

  type = "correlation" AND demean = TRUE

$$\hat{\rho}(k) = \frac{\sum_{t=1}^{n-k} (x_t - \bar{x})\,(x_{t+k} - \bar{x})}{\sum_{t=1}^{n} (x_t - \bar{x})^2} \quad \forall\, k = 0, 1, 2, \ldots, d$$

  type = "correlation" AND demean = FALSE

$$\hat{\rho}(k) = \frac{\sum_{t=1}^{n-k} x_t\, x_{t+k}}{\sum_{t=1}^{n} x_t^2} \quad \forall\, k = 0, 1, 2, \ldots, d$$

  type = "covariance" AND demean = TRUE

$$\hat{\gamma}(k) = \frac{1}{n} \sum_{t=1}^{n-k} (x_t - \bar{x})\,(x_{t+k} - \bar{x}) \quad \forall\, k = 0, 1, 2, \ldots, d$$

  type = "covariance" AND demean = FALSE

$$\hat{\gamma}(k) = \frac{1}{n} \sum_{t=1}^{n-k} x_t \, x_{t+k} \quad \forall \, k = 0, 1, 2, \ldots, d$$

`type = "partial"`

$$\hat{\pi}(k) = \frac{\begin{vmatrix} 1 & \hat{\rho}(1) & \hat{\rho}(2) & \ldots & \hat{\rho}(1) \\ \hat{\rho}(1) & 1 & \hat{\rho}(1) & \ldots & \hat{\rho}(2) \\ \hat{\rho}(2) & \hat{\rho}(1) & 1 & \ldots & \hat{\rho}(3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{\rho}(k-1) & \hat{\rho}(k-2) & \hat{\rho}(k-3) & \ldots & \hat{\rho}(k) \end{vmatrix}}{\begin{vmatrix} 1 & \hat{\rho}(1) & \hat{\rho}(2) & \ldots & \hat{\rho}(k-1) \\ \hat{\rho}(1) & 1 & \hat{\rho}(1) & \ldots & \hat{\rho}(k-2) \\ \hat{\rho}(2) & \hat{\rho}(1) & 1 & \ldots & \hat{\rho}(k-3) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{\rho}(k-1) & \hat{\rho}(k-2) & \hat{\rho}(k-3) & \ldots & 1 \end{vmatrix}} \quad \forall \, k = 1, 2, \ldots, d$$

`n.used`

$$n$$

`lag`

$$d$$

- **Examples:**

```
> x <- c(1, 2, 7, 3, 5, 2, 0, 1, 4, 5)
> n <- 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.max = d, type = "correlation", demean = TRUE, plot = FALSE)$acf[d +
+     1]

[1] -0.3409091

> x <- c(1, 2, 7, 3, 5, 2, 0, 1, 4, 5)
> n <- 10
> d <- 4
> sum((x[1:(n - d)]) * (x[(d + 1):n]))/(sum(x^2))

[1] 0.3134328

> acf(x, lag.max = d, type = "correlation", demean = FALSE, plot = FALSE)$acf[d +
+     1]

[1] 0.3134328

> x <- c(1, 2, 7, 3, 5, 2, 0, 1, 4, 5)
> n <- 10
> d <- 4
> sum((x[1:(n - d)] - mean(x)) * (x[(d + 1):n] - mean(x)))/n

[1] -1.5

> acf(x, lag.max = d, type = "covariance", demean = TRUE, plot = FALSE)$acf[d +
+     1]

[1] -1.5
```

```
> x <- c(1, 2, 7, 3, 5, 2, 0, 1, 4, 5)
> n <- 10
> d <- 4
> sum((x[1:(n - d)]) * (x[(d + 1):n]))/n

[1] 4.2

> acf(x, lag.max = d, type = "covariance", demean = FALSE, plot = FALSE)$acf[d +
+     1]

[1] 4.2
```

pacf()

- **Package:** stats
- **Input:**

  x vettore numerico di dimensione n

  lag.max il valore d del ritardo

  demean = TRUE / FALSE centratura

  plot = FALSE
- **Description:** 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 = TRUE

$$\hat{\rho}(k) = \frac{\sum_{t=1}^{n-k} (x_t - \bar{x})\,(x_{t+k} - \bar{x})}{\sum_{t=1}^{n} (x_t - \bar{x})^2} \quad \forall\, k = 0, 1, 2, \dots, d$$

demean = FALSE

$$\hat{\rho}(k) = \frac{\sum_{t=1}^{n-k} x_t\, x_{t+k}}{\sum_{t=1}^{n} x_t^2} \quad \forall\, k = 0, 1, 2, \dots, d$$

n.used

$$n$$

lag

$$d$$

- **Examples:**

```
> x <- c(1, 2, 7, 3, 5, 2, 0, 1, 4, 5)
> n <- 10
> d <- 4
> pacf(x, lag.max = d, demean = TRUE, plot = FALSE)

Partial autocorrelations of series 'x', by lag

     1      2      3      4
 0.114 -0.266 -0.349 -0.417
```

3.24 Valori mancanti

is.na()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

- **Description:** rileva la presenza di valori NA e NaN
- **Examples:**

```
> x <- c(1.3, 1, 2, 3.4, 3.4, 5.7, NA, 3.8)
> is.na(x)

[1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE

> x <- c(1.3, NaN, 2, 3.4, 3.4, 5.7, NA, 3.8)
> is.na(x)

[1] FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE

> x <- c(1, 2, NA, 4, 5.6, NaN, 1.2, 4, 4.4)
> x[!is.na(x)]

[1] 1.0 2.0 4.0 5.6 1.2 4.0 4.4

> x <- c(3, 4, NA, 5)
> mean(x)

[1] NA

> mean(x[!is.na(x)])

[1] 4
```

is.nan()

- **Package:** base
- **Input:**

  x vettore numerico di dimensione n

- **Description:** rileva la presenza di valori NaN
- **Examples:**

```
> x <- c(1.3, 1, 2, 3.4, 3.4, 5.7, NA, 3.8)
> is.nan(x)

[1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE


> x <- c(1.3, NaN, 2, 3.4, 3.4, 5.7, NA, 3.8)
> is.nan(x)

[1] FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE


> x <- c(1, 2, NA, 4, 5.6, NaN, 1.2, 4, 4.4)
> x[!is.nan(x)]

[1] 1.0 2.0  NA 4.0 5.6 1.2 4.0 4.4
```

na.omit()

- **Package:** stats
- **Input:**

  x vettore numerico di dimensione n

- **Description:** elimina i valori NA e NaN
- **Examples:**

```
> x <- c(1.3, 1, 2, 3.4, 3.4, 5.7, NA, 3.8)
> na.omit(x)

[1] 1.3 1.0 2.0 3.4 3.4 5.7 3.8
attr(,"na.action")
[1] 7
attr(,"class")
[1] "omit"


> x <- c(1.3, NaN, 2, 3.4, 3.4, 5.7, NA, 3.8)
> na.omit(x)

[1] 1.3 2.0 3.4 3.4 5.7 3.8
attr(,"na.action")
[1] 2 7
attr(,"class")
[1] "omit"
```

3.25 Miscellaneous

sample()

- **Package:** fUtilities
- **Input:**

  x vettore alfanumerico di dimensione n

  size ampiezza campionaria

  replace = TRUE / FALSE estrazione con oppure senza ripetizione

  prob vettore di probabilità

- **Description:** estrazione campionaria
- **Examples:**

```
> x <- c("A", "B")
> n <- 2
> sample(x, size = 10, replace = TRUE, prob = rep(1/n, times = n))

 [1] "B" "A" "B" "A" "B" "A" "B" "B" "B" "B"

> x <- c(0, 1)
> n <- 2
> sample(x, size = 5, replace = TRUE, prob = rep(1/n, times = n))

[1] 1 0 1 0 1

> x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
> n <- 10
> sample(x, size = 3, replace = FALSE, prob = rep(1/n, times = n))

[1] 9 2 4
```

nsize()

- **Package:** BSDA
- **Input:**

  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

- **Description:** dimensione campionaria dato il margine di errore E
- **Formula:**

type = "mu"

$$n = \left\lceil \left(z_{1-\alpha\,/\,2}\, \sigma_x \right) / \, E)^2 \right\rceil$$

type = "pi"

$$n = \left\lceil p\,(1-p)\,(z_{1-\alpha\,/\,2}\,/\,E)^2 \right\rceil$$

- **Examples:**

```
> 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 .
```

ic.var()

- **Package:** labstatR
- **Input:**

  x vettore numerico di dimensione n

  conf.level livello di confidenza $1 - \alpha$

- **Description:** intervallo di confidenza Chi-Quadrato per la varianza incognita
- **Formula:**

$$\frac{(n-1)\, s_x^2}{\chi^2_{1-\alpha/2,\, n-1}} \quad \frac{(n-1)\, s_x^2}{\chi^2_{\alpha/2,\, n-1}}$$

- **Examples:**

```
> x <- c(1.2, 3.4, 4.2, 12.4, 13.4, 17.3, 18.1)
> n <- 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 <- c(1, 2, 3, 4, 5.6, 7.4, 1.2, 4, 4.4)
> n <- 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
```

sweep()

- **Package:** base
- **Input:**

  x matrice di dimensione $n \times k$

  MARGIN = 1 / 2 righe oppure colonne

  STATS statistica da calcolare su ogni riga (colonna) della matrice x

  FUN operazione da compiere tra ogni riga (colonna) e la statistica riassuntiva di riga (colonna)

- **Description:** operazioni da compiere su ogni riga (colonna) della matrice x
- **Examples:**

```
> X1 <- c(1.2, 3.4, 5.6)
> X2 <- c(7.5, 6.7, 8.4)
> X3 <- c(4.3, 3.2, 3.2)
> x <- cbind(X1, X2, X3)
> mediecolonna <- apply(x, MARGIN = 2, FUN = mean)
> mediecolonna

      X1       X2       X3
3.400000 7.533333 3.566667

> sweep(x, MARGIN = 2, STATS = mediecolonna, FUN = "-")

       X1          X2         X3
[1,] -2.2 -0.03333333  0.7333333
[2,]  0.0 -0.83333333 -0.3666667
[3,]  2.2  0.86666667 -0.3666667

> X1 <- c(1.2, 3.4, 5.6)
> X2 <- c(7.5, 6.7, 8.4)
> X3 <- c(4.3, 3.2, 3.2)
> x <- cbind(X1, X2, X3)
> medieriga <- apply(x, MARGIN = 1, FUN = mean)
> medieriga

[1] 4.333333 4.433333 5.733333

> sweep(x, MARGIN = 1, STATS = medieriga, FUN = "-")

            X1       X2          X3
[1,] -3.1333333 3.166667 -0.03333333
[2,] -1.0333333 2.266667 -1.23333333
[3,] -0.1333333 2.666667 -2.53333333
```

set.seed()

- **Package:** base
- **Input:**

  seed seme

- **Description:** fissa un seme per rendere riproducibili i risultati di un'estrazione
- **Examples:**

```
> set.seed(seed = 100)
> rnorm(1)
```

```
[1] -0.5021924

> rnorm(1)

[1] 0.1315312

> rnorm(1)

[1] -0.07891709

> rnorm(1)

[1] 0.8867848

> set.seed(seed = 100)
> rnorm(1)

[1] -0.5021924

> rnorm(1)

[1] 0.1315312
```

simple.z.test()

- **Package:** UsingR
- **Input:**

  `x` vettore numerico di dimensione n

  `sigma` valore di σ_x

  `conf.level` livello di confidenza $1 - \alpha$

- **Description:** intervallo di confidenza per la media incognita a livello $1 - \alpha$
- **Formula:**

$$\bar{x} \mp z_{1-\alpha\,/\,2}\,\sigma_x\,/\,\sqrt{n}$$

- **Example:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1, 6.7, 7.6, 6.8)
> xmedio <- mean(x)
> xmedio

[1] 7.018182

> sigmax <- 1.2
> alpha <- 0.05
> n <- 11
> 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

> simple.z.test(x, sigma = 1.2, conf.level = 0.95)

[1] 6.309040 7.727323
```

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> xmedio <- mean(x)
> xmedio

[1] 4.68

> sigmax <- 1.45
> alpha <- 0.05
> n <- 5
> 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

> simple.z.test(x, sigma = 1.45, conf.level = 0.95)

[1] 3.409042 5.950958
```

median.test()

- **Package:** formularioR
- **Input:**

  x vettore numerico di dimensione n

  m0 valore $Q_{0.5}(x)$ della mediana
- **Description:** verifica di ipotesi per la mediana
- **Formula:**

$$2 \min \left(P(X \leq v), P(X \geq v)\right)$$

$$\text{dove} \quad X \sim Binomiale(n,\, p_0) \qquad v \,=\, \#\left(x_i < Q_{0.5}(x) \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n\right)$$

- **Example:**

```
> x <- c(1, 2, 8, 12, 12, 17, 25, 52)
> n <- 8
> m0 <- 12
> v <- sum(x < 12)
> v

[1] 3

> 2 * min(pbinom(q = v, size = 8, prob = 0.5), 1 - pbinom(q = v -
+     1, size = 8, prob = 0.5))

[1] 0.7265625

> median.test(x, m0 = 12)

[1] 0.7265625

> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1, 6.7, 7.6, 6.8)
> n <- 11
> m0 <- 6.6
> v <- sum(x < 6.6)
> v
```

```
[1] 2

> 2 * min(pbinom(q = v, size = 11, prob = 0.5), 1 - pbinom(q = v -
+     1, size = 11, prob = 0.5))

[1] 0.06542969

> median.test(x, m0 = 6.6)

[1] 0.06542969
```

Capitolo 4

Analisi Componenti Principali (ACP)

4.1 ACP con matrice di covarianza di popolazione

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 dei dati centrata di dimensione $n \times k$: Z
- elemento di riga i e colonna j della matrice dei dati 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 $k \times k$: $S = \frac{Z^T Z}{n} = \Gamma D \Gamma^T$
- 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: $x_j = Z \Gamma^j \quad \forall j = 1, 2, \ldots, k$
- scarto quadratico medio della j-esima componente principale:
$\sigma_{x_j} = \sqrt{\lambda_{(k-j+1)}} \quad \forall j = 1, 2, \ldots, k$
- problema di ottimo vincolato:
$x_j = Z \gamma_j \quad \forall j = 1, 2, \ldots, k$
$\sigma^2_{x_j} = \frac{x_j^T x_j}{n} = \frac{(Z \gamma_j)^T (Z \gamma_j)}{n} = \gamma_j^T \frac{Z^T Z}{n} \gamma_j = \gamma_j^T S \gamma_j \quad \forall j = 1, 2, \ldots, k$
$\max_{\gamma_j^T \gamma_j = 1} \sigma^2_{x_j} = \max_{\gamma_j^T \gamma_j = 1} \gamma_j^T S \gamma_j = \lambda_{(k-j+1)} \quad \forall j = 1, 2, \ldots, k$

princomp()

- **Package:** `stats`
- **Input:**

  `W` matrice dei dati

- **Output:**

  `sdev` scarto quadratico medio delle componenti principali

  `center` media di colonna della matrice W

  `n.obs` dimensione campionaria

  `scores` componenti principali

- **Formula:**

  `sdev`

$$\sigma_{x_j} \quad \forall j = 1, 2, \ldots, k$$

center

$$\bar{w}_j \quad \forall\, j = 1, 2, \ldots, k$$

n.obs

$$n$$

scores

$$x_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> w1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> w2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> w3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> W <- cbind(w1, w2, w3)
> W

      w1  w2   w3
[1,] 1.1 1.2 1.40
[2,] 2.3 3.4 5.60
[3,] 4.5 5.6 7.56
[4,] 6.7 7.5 6.00
[5,] 8.9 7.5 5.40
[6,] 3.4 6.7 6.60
[7,] 5.6 8.6 8.70
[8,] 6.7 7.6 8.70

> res <- princomp(W)
> n <- 8
> k <- 3
> Z <- scale(W, scale = FALSE)
> colnames(Z) <- c("z1", "z2", "z3")
> Z

       z1      z2     z3
[1,] -3.8 -4.8125 -4.845
[2,] -2.6 -2.6125 -0.645
[3,] -0.4 -0.4125  1.315
[4,]  1.8  1.4875 -0.245
[5,]  4.0  1.4875 -0.845
[6,] -1.5  0.6875  0.355
[7,]  0.7  2.5875  2.455
[8,]  1.8  1.5875  2.455
attr(,"scaled:center")
    w1     w2     w3
4.9000 6.0125 6.2450

> S <- (1/n) * t(Z) %*% Z
> dimnames(S) <- list(NULL, NULL)
> S

         [,1]     [,2]     [,3]
[1,] 5.82250 4.688750 2.668250
[2,] 4.68875 5.533594 4.166437
[3,] 2.66825 4.166437 4.821675

> sdev <- sqrt(eigen(S)$values)
> names(sdev) <- c("Comp.1", "Comp.2", "Comp.3")
> sdev

   Comp.1    Comp.2    Comp.3
3.6303620 1.6179210 0.6169052

> res$sdev
```

```
   Comp.1    Comp.2    Comp.3
3.6303620 1.6179210 0.6169052

> center <- apply(W, MARGIN = 2, FUN = mean)
> center

    w1     w2     w3
4.9000 6.0125 6.2450

> res$center

    w1     w2     w3
4.9000 6.0125 6.2450

> n

[1] 8

> res$n.obs

[1] 8

> D <- diag(eigen(S)$values)
> D

         [,1]     [,2]      [,3]
[1,] 13.17953 0.000000 0.0000000
[2,]  0.00000 2.617668 0.0000000
[3,]  0.00000 0.000000 0.3805721

> GAMMA <- eigen(S)$vectors
> GAMMA

          [,1]        [,2]       [,3]
[1,] 0.5867813  0.68021602  0.4393107
[2,] 0.6341906 -0.04872184 -0.7716401
[3,] 0.5034779 -0.73139069  0.4599757

> scores <- Z %*% GAMMA
> colnames(scores) <- c("Comp.1", "Comp.2", "Comp.3")
> scores

         Comp.1     Comp.2     Comp.3
[1,] -7.7211617  1.1932409 -0.1844450
[2,] -3.5071975 -1.1695288  0.5770175
[3,]  0.1657573 -1.2137674  0.7474453
[4,]  1.8762127  1.3311058 -0.4697494
[5,]  2.8650447  3.2664155  0.2207489
[6,] -0.2654312 -1.3134640 -1.0261773
[7,]  3.2877534 -1.4454807 -0.5598609
[8,]  3.2990222 -0.6485212  0.6950210

> res$scores

         Comp.1     Comp.2     Comp.3
[1,]  7.7211617  1.1932409 -0.1844450
[2,]  3.5071975 -1.1695288  0.5770175
[3,] -0.1657573 -1.2137674  0.7474453
[4,] -1.8762127  1.3311058 -0.4697494
[5,] -2.8650447  3.2664155  0.2207489
[6,]  0.2654312 -1.3134640 -1.0261773
[7,] -3.2877534 -1.4454807 -0.5598609
[8,] -3.2990222 -0.6485212  0.6950210
```

4.2 ACP con matrice di covarianza campionaria

Simbologia

- matrice dei dati di dimensione $n \times k$ le cui colonne corrispondono ai vettori numerici $w_1, w_2, \dots, w_k$: W
- media di colonna della matrice dei dati: $\bar{w}_j \quad \forall j = 1, 2, \dots, k$
- matrice dei dati centrata di dimensione $n \times k$: Z
- elemento di riga i e colonna j della matrice dei dati centrata:
  $z_{ij} = w_{ij} - \bar{w}_j \quad \forall i = 1, 2, \dots, n \quad \forall j = 1, 2, \dots, k$
- 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: $\Gamma^j \quad \forall j = 1, 2, \dots, k$
- matrice diagonale degli autovalori di dimensione $k \times k$: $D = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_k)$
- componente principale j-esima: $x_j = Z \Gamma^j \quad \forall j = 1, 2, \dots, k$
- deviazione standard della j-esima componente principale:
  $s_{x_j} = \sqrt{\lambda_{(k-j+1)}} \quad \forall j = 1, 2, \dots, k$
- problema di ottimo vincolato:
  $x_j = Z \gamma_j \quad \forall j = 1, 2, \dots, k$
  $s_{x_j}^2 = \frac{x_j^T x_j}{n-1} = \frac{(Z\gamma_j)^T (Z\gamma_j)}{n-1} = \gamma_j^T \frac{Z^T Z}{n-1} \gamma_j = \gamma_j^T S \gamma_j \quad \forall j = 1, 2, \dots, k$
  $\max_{\gamma_j^T \gamma_j = 1} s_{x_j}^2 = \max_{\gamma_j^T \gamma_j = 1} \gamma_j^T S \gamma_j = \lambda_{(k-j+1)} \quad \forall j = 1, 2, \dots, k$

prcomp()

- **Package:** `stats`
- **Input:**

  `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`

  $$s_{x_j} \quad \forall j = 1, 2, \dots, k$$

  `rotation`

  $$\Gamma$$

  `center`

  $$\bar{w}_j \quad \forall j = 1, 2, \dots, k$$

  `x`

  $$x_j \quad \forall j = 1, 2, \dots, k$$

- **Examples:**

```
> w1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> w2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> w3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> W <- cbind(w1, w2, w3)
> W
```

```
      w1  w2   w3
[1,] 1.1 1.2 1.40
[2,] 2.3 3.4 5.60
[3,] 4.5 5.6 7.56
[4,] 6.7 7.5 6.00
[5,] 8.9 7.5 5.40
[6,] 3.4 6.7 6.60
[7,] 5.6 8.6 8.70
[8,] 6.7 7.6 8.70
```

```
> res <- prcomp(W)
> n <- 8
> k <- 3
> Z <- scale(W, scale = FALSE)
> colnames(Z) <- c("z1", "z2", "z3")
> Z
```

```
       z1      z2     z3
[1,] -3.8 -4.8125 -4.845
[2,] -2.6 -2.6125 -0.645
[3,] -0.4 -0.4125  1.315
[4,]  1.8  1.4875 -0.245
[5,]  4.0  1.4875 -0.845
[6,] -1.5  0.6875  0.355
[7,]  0.7  2.5875  2.455
[8,]  1.8  1.5875  2.455
attr(,"scaled:center")
    w1     w2     w3
4.9000 6.0125 6.2450
```

```
> S <- (1/(n - 1)) * t(Z) %*% Z
> dimnames(S) <- list(NULL, NULL)
> S
```

```
         [,1]     [,2]     [,3]
[1,] 6.654286 5.358571 3.049429
[2,] 5.358571 6.324107 4.761643
[3,] 3.049429 4.761643 5.510486
```

```
> sdev <- sqrt(eigen(S)$values)
> sdev
```

```
[1] 3.8810202 1.7296303 0.6594994
```

```
> res$sdev
```

```
[1] 3.8810202 1.7296303 0.6594994
```

```
> GAMMA <- eigen(S)$vectors
> dimnames(GAMMA) <- list(c("w1", "w2", "w3"), c("PC1", "PC2",
+     "PC3"))
> GAMMA
```

```
          PC1         PC2        PC3
w1 -0.5867813 -0.68021602  0.4393107
w2 -0.6341906  0.04872184 -0.7716401
w3 -0.5034779  0.73139069  0.4599757
```

```
> res$rotation
```

```
          PC1         PC2        PC3
w1 0.5867813  0.68021602 -0.4393107
w2 0.6341906 -0.04872184  0.7716401
w3 0.5034779 -0.73139069 -0.4599757
```

```
> center <- apply(W, MARGIN = 2, FUN = mean)
> center
```

```
    w1     w2     w3
4.9000 6.0125 6.2450
```

```
> res$center
```

```
    w1     w2     w3
4.9000 6.0125 6.2450
```

```
> D <- diag(eigen(S)$values)
> D
```

```
         [,1]     [,2]      [,3]
[1,] 15.06232 0.000000 0.0000000
[2,]  0.00000 2.991621 0.0000000
[3,]  0.00000 0.000000 0.4349395
```

```
> scores <- Z %*% GAMMA
> colnames(scores) <- c("PC1", "PC2", "PC3")
> scores
```

```
            PC1        PC2        PC3
[1,]  7.7211617 -1.1932409 -0.1844450
[2,]  3.5071975  1.1695288  0.5770175
[3,] -0.1657573  1.2137674  0.7474453
[4,] -1.8762127 -1.3311058 -0.4697494
[5,] -2.8650447 -3.2664155  0.2207489
[6,]  0.2654312  1.3134640 -1.0261773
[7,] -3.2877534  1.4454807 -0.5598609
[8,] -3.2990222  0.6485212  0.6950210
```

```
> res$x
```

```
            PC1        PC2        PC3
[1,] -7.7211617  1.1932409  0.1844450
[2,] -3.5071975 -1.1695288 -0.5770175
[3,]  0.1657573 -1.2137674 -0.7474453
[4,]  1.8762127  1.3311058  0.4697494
[5,]  2.8650447  3.2664155 -0.2207489
[6,] -0.2654312 -1.3134640  1.0261773
[7,]  3.2877534 -1.4454807  0.5598609
[8,]  3.2990222 -0.6485212 -0.6950210
```

summary()

- **Package:** base
- **Input:**

  `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 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`

$$s_{x_j} \qquad \frac{\lambda_{(k-j+1)}}{\sum_{i=1}^{k} \lambda_i} \qquad \frac{\sum_{j=1}^{l} \lambda_{(k-j+1)}}{\sum_{i=1}^{k} \lambda_i} \qquad \forall\, j, l = 1, 2, \ldots, k$$

- **Examples:**

```
> w1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> w2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> w3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> W <- cbind(w1, w2, w3)
> W

      w1  w2   w3
[1,] 1.1 1.2 1.40
[2,] 2.3 3.4 5.60
[3,] 4.5 5.6 7.56
[4,] 6.7 7.5 6.00
[5,] 8.9 7.5 5.40
[6,] 3.4 6.7 6.60
[7,] 5.6 8.6 8.70
[8,] 6.7 7.6 8.70

> res <- summary(object = prcomp(W))
> n <- 8
> k <- 3
> Z <- scale(W, scale = FALSE)
> colnames(Z) <- c("z1", "z2", "z3")
> Z

       z1      z2     z3
[1,] -3.8 -4.8125 -4.845
[2,] -2.6 -2.6125 -0.645
[3,] -0.4 -0.4125  1.315
[4,]  1.8  1.4875 -0.245
[5,]  4.0  1.4875 -0.845
[6,] -1.5  0.6875  0.355
[7,]  0.7  2.5875  2.455
[8,]  1.8  1.5875  2.455
attr(,"scaled:center")
    w1     w2     w3
4.9000 6.0125 6.2450
```

```
> S <- (1/(n - 1)) * t(Z) %*% Z
> dimnames(S) <- list(NULL, NULL)
> S

         [,1]     [,2]     [,3]
[1,] 6.654286 5.358571 3.049429
[2,] 5.358571 6.324107 4.761643
[3,] 3.049429 4.761643 5.510486

> sdev <- sqrt(eigen(S)$values)
> sdev

[1] 3.8810202 1.7296303 0.6594994

> res$sdev

[1] 3.8810202 1.7296303 0.6594994

> GAMMA <- eigen(S)$vectors
> GAMMA

           [,1]        [,2]       [,3]
[1,] -0.5867813 -0.68021602  0.4393107
[2,] -0.6341906  0.04872184 -0.7716401
[3,] -0.5034779  0.73139069  0.4599757

> res$rotation

         PC1         PC2        PC3
w1 0.5867813  0.68021602 -0.4393107
w2 0.6341906 -0.04872184  0.7716401
w3 0.5034779 -0.73139069 -0.4599757

> center <- apply(W, MARGIN = 2, FUN = mean)
> center

    w1     w2     w3
4.9000 6.0125 6.2450

> res$center

    w1     w2     w3
4.9000 6.0125 6.2450

> D <- diag(eigen(S)$values)
> D

         [,1]     [,2]      [,3]
[1,] 15.06232 0.000000 0.0000000
[2,]  0.00000 2.991621 0.0000000
[3,]  0.00000 0.000000 0.4349395

> x <- Z %*% GAMMA
> colnames(x) <- c("PC1", "PC2", "PC3")
> x
```

```
            PC1        PC2        PC3
[1,]  7.7211617 -1.1932409 -0.1844450
[2,]  3.5071975  1.1695288  0.5770175
[3,] -0.1657573  1.2137674  0.7474453
[4,] -1.8762127 -1.3311058 -0.4697494
[5,] -2.8650447 -3.2664155  0.2207489
[6,]  0.2654312  1.3134640 -1.0261773
[7,] -3.2877534  1.4454807 -0.5598609
[8,] -3.2990222  0.6485212  0.6950210
```

```
> res$x
```

```
            PC1        PC2        PC3
[1,] -7.7211617  1.1932409  0.1844450
[2,] -3.5071975 -1.1695288 -0.5770175
[3,]  0.1657573 -1.2137674 -0.7474453
[4,]  1.8762127  1.3311058  0.4697494
[5,]  2.8650447  3.2664155 -0.2207489
[6,] -0.2654312 -1.3134640  1.0261773
[7,]  3.2877534 -1.4454807  0.5598609
[8,]  3.2990222 -0.6485212 -0.6950210
```

```
> lambda <- sdev^2
> importance <- rbind(sdev, lambda/sum(lambda), cumsum(lambda)/sum(lambda))
> dimnames(importance) <- list(c("Standard deviation", "Proportion of Variance",
+     "Cumulative Proportion"), c("PC1", "PC2", "PC3"))
> importance
```

```
                             PC1       PC2        PC3
Standard deviation     3.8810202 1.7296303 0.65949942
Proportion of Variance 0.8146691 0.1618065 0.02352438
Cumulative Proportion  0.8146691 0.9764756 1.00000000
```

```
> res$importance
```

```
                           PC1      PC2       PC3
Standard deviation     3.88102 1.729630 0.6594994
Proportion of Variance 0.81467 0.161810 0.0235200
Cumulative Proportion  0.81467 0.976480 1.0000000
```

4.3 ACP con matrice di correlazione di popolazione

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:
  $\sigma^2_{w_j} = n^{-1} (w_j - \bar{w}_j)^T (w_j - \bar{w}_j) \quad \forall j = 1, 2, \ldots, k$
- matrice dei dati standardizzata di dimensione $n \times k$: Z
- elemento di riga i e colonna j della matrice dei dati standardizzata:
  $z_{ij} = (w_{ij} - \bar{w}_j) / \sigma_{w_j} \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$
- matrice di correlazione di dimensione $k \times k$: $R = \frac{Z^T Z}{n} = \Gamma D \Gamma^T$
- 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: $x_j = Z\,\Gamma^j \quad \forall\, j = 1, 2, \ldots, k$
- scarto quadratico medio della j-esima componente principale:
  $\sigma_{x_j} = \sqrt{\lambda_{(k-j+1)}} \quad \forall\, j = 1, 2, \ldots, k$
- problema di ottimo vincolato:
  $x_j = Z\,\gamma_j \quad \forall\, j = 1, 2, \ldots, k$
  $\sigma^2_{x_j} = \frac{x_j^T\,x_j}{n} = \frac{(Z\,\gamma_j)^T\,(Z\,\gamma_j)}{n} = \gamma_j^T\,\frac{Z^T\,Z}{n}\,\gamma_j = \gamma_j^T\,R\,\gamma_j \quad \forall\, j = 1, 2, \ldots, k$
  $\max_{\gamma_j^T\,\gamma_j = 1}\,\sigma^2_{x_j} = \max_{\gamma_j^T\,\gamma_j = 1}\,\gamma_j^T\,R\,\gamma_j = \lambda_{(k-j+1)} \quad \forall\, j = 1, 2, \ldots, k$

princomp()

- **Package:** stats
- **Input:**

  `W` matrice dei dati

  `cor = TRUE` matrice di correlazione

- **Output:**

  `sdev` scarto quadratico medio delle componenti principali

  `center` media di colonna della matrice W

  `scale` scarto quadratico medio di colonna della matrice W

  `n.obs` dimensione campionaria

  `scores` componenti principali

- **Formula:**

  `sdev`
  $$\sigma_{x_j} \quad \forall\, j = 1, 2, \ldots, k$$

  `center`
  $$\bar{w}_j \quad \forall\, j = 1, 2, \ldots, k$$

  `scale`
  $$\sigma_{w_j} \quad \forall\, j = 1, 2, \ldots, k$$

  `n.obs`
  $$n$$

  `scores`
  $$x_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> w1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> w2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> w3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> W <- cbind(w1, w2, w3)
> W
```

```
      w1  w2   w3
[1,] 1.1 1.2 1.40
[2,] 2.3 3.4 5.60
[3,] 4.5 5.6 7.56
[4,] 6.7 7.5 6.00
[5,] 8.9 7.5 5.40
[6,] 3.4 6.7 6.60
[7,] 5.6 8.6 8.70
[8,] 6.7 7.6 8.70
```

```
> res <- princomp(W, cor = TRUE)
> n <- 8
> k <- 3
> sigma <- function(x) sqrt((length(x) - 1) * var(x)/length(x))
> Z <- sweep(W, 2, apply(W, MARGIN = 2, FUN = mean)) %*% diag(1/apply(W,
+     MARGIN = 2, FUN = sigma))
> colnames(Z) <- c("z1", "z2", "z3")
> Z

             z1         z2         z3
[1,] -1.5748125 -2.0458185 -2.2064537
[2,] -1.0775033 -1.1105872 -0.2937384
[3,] -0.1657697 -0.1753559  0.5988620
[4,]  0.7459638  0.6323439 -0.1115751
[5,]  1.6576973  0.6323439 -0.3848201
[6,] -0.6216365  0.2922598  0.1616700
[7,]  0.2900970  1.0999596  1.1180276
[8,]  0.7459638  0.6748544  1.1180276

> R <- (1/n) * t(Z) %*% Z
> dimnames(R) <- list(NULL, NULL)
> R

          [,1]      [,2]      [,3]
[1,] 1.0000000 0.8260355 0.5035850
[2,] 0.8260355 1.0000000 0.8066075
[3,] 0.5035850 0.8066075 1.0000000

> sdev <- sqrt(eigen(R)$values)
> names(sdev) <- c("Comp.1", "Comp.2", "Comp.3")
> sdev

   Comp.1    Comp.2    Comp.3
1.5599434 0.7047305 0.2644457

> res$sdev

   Comp.1    Comp.2    Comp.3
1.5599434 0.7047305 0.2644457

> center <- apply(W, MARGIN = 2, FUN = mean)
> center

    w1     w2     w3
4.9000 6.0125 6.2450

> res$center

    w1     w2     w3
4.9000 6.0125 6.2450

> scale <- apply(W, MARGIN = 2, FUN = sigma)
> scale

      w1       w2       w3
2.412986 2.352359 2.195831

> res$scale
```

```
      w1       w2       w3
2.412986 2.352359 2.195831
```

```
> n
```

```
[1] 8
```

```
> res$n.obs
```

```
[1] 8
```

```
> D <- diag(eigen(R)$values)
> D
```

```
         [,1]      [,2]      [,3]
[1,] 2.433423 0.0000000 0.0000000
[2,] 0.000000 0.4966451 0.0000000
[3,] 0.000000 0.0000000 0.0699315
```

```
> GAMMA <- eigen(R)$vectors
> GAMMA
```

```
           [,1]        [,2]       [,3]
[1,] -0.5538345 -0.69330367  0.4610828
[2,] -0.6272670 -0.01674325 -0.7786242
[3,] -0.5475431  0.72045103  0.4256136
```

```
> scores <- Z %*% GAMMA
> colnames(scores) <- c("Comp.1", "Comp.2", "Comp.3")
> scores
```

```
          Comp.1     Comp.2      Comp.3
[1,]  3.36358843 -0.4635649 -0.07229172
[2,]  1.45422766  0.5540077  0.24289279
[3,] -0.12609881  0.5493156  0.31498656
[4,] -0.74869682 -0.6081513 -0.19589504
[5,] -1.10403287 -1.4371192  0.10819286
[6,]  0.07243752  0.5425648 -0.44537755
[7,] -1.46280241  0.5859419 -0.24684871
[8,] -1.44862269  0.2770054  0.29434081
```

```
> res$scores
```

```
          Comp.1     Comp.2      Comp.3
[1,]  3.36358843 -0.4635649 -0.07229172
[2,]  1.45422766  0.5540077  0.24289279
[3,] -0.12609881  0.5493156  0.31498656
[4,] -0.74869682 -0.6081513 -0.19589504
[5,] -1.10403287 -1.4371192  0.10819286
[6,]  0.07243752  0.5425648 -0.44537755
[7,] -1.46280241  0.5859419 -0.24684871
[8,] -1.44862269  0.2770054  0.29434081
```

4.4 ACP con matrice di correlazione campionaria

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:
  $s_{w_j}^2 = (n-1)^{-1} (w_j - \bar{w}_j)^T (w_j - \bar{w}_j) \quad \forall j = 1, 2, \ldots, k$
- matrice dei dati standardizzata di dimensione $n \times k$: Z
- elemento di riga i e colonna j della matrice dei dati 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:
  $x_j = Z \gamma_j \quad \forall j = 1, 2, \ldots, k$
  $s_{x_j}^2 = \frac{x_j^T x_j}{n-1} = \frac{(Z \gamma_j)^T (Z \gamma_j)}{n-1} = \gamma_j^T \frac{Z^T Z}{n-1} \gamma_j = \gamma_j^T R \gamma_j \quad \forall j = 1, 2, \ldots, k$
  $\max_{\gamma_j^T \gamma_j = 1} s_{x_j}^2 = \max_{\gamma_j^T \gamma_j = 1} \gamma_j^T R \gamma_j = \lambda_{(k-j+1)} \quad \forall j = 1, 2, \ldots, k$

prcomp()

- **Package:** `stats`
- **Input:**

  `W` matrice dei dati

  `scale. = TRUE` matrice di correlazione
- **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`
  $$x_j \quad \forall j = 1, 2, \ldots, k$$
- **Examples:**

```
> w1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> w2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> w3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> W <- cbind(w1, w2, w3)
> W

      w1  w2   w3
[1,] 1.1 1.2 1.40
[2,] 2.3 3.4 5.60
[3,] 4.5 5.6 7.56
[4,] 6.7 7.5 6.00
[5,] 8.9 7.5 5.40
[6,] 3.4 6.7 6.60
[7,] 5.6 8.6 8.70
[8,] 6.7 7.6 8.70

> res <- prcomp(W, scale. = TRUE)
> n <- 8
> k <- 3
> Z <- scale(W, scale = TRUE)
> colnames(Z) <- c("z1", "z2", "z3")
> Z

             z1         z2         z3
[1,] -1.4731022 -1.9136880 -2.0639484
[2,] -1.0079120 -1.0388592 -0.2747671
[3,] -0.1550634 -0.1640304  0.5601841
[4,]  0.6977852  0.5915036 -0.1043689
[5,]  1.5506339  0.5915036 -0.3599662
[6,] -0.5814877  0.2733840  0.1512284
[7,]  0.2713609  1.0289180  1.0458191
[8,]  0.6977852  0.6312685  1.0458191
attr(,"scaled:center")
    w1     w2     w3
4.9000 6.0125 6.2450
attr(,"scaled:scale")
      w1       w2       w3
2.579590 2.514778 2.347442

> R <- (1/(n - 1)) * t(Z) %*% Z
> dimnames(R) <- list(NULL, NULL)
> R

          [,1]      [,2]      [,3]
[1,] 1.0000000 0.8260355 0.5035850
[2,] 0.8260355 1.0000000 0.8066075
[3,] 0.5035850 0.8066075 1.0000000

> sdev <- sqrt(eigen(R)$values)
> sdev

[1] 1.5599434 0.7047305 0.2644457

> res$sdev

[1] 1.5599434 0.7047305 0.2644457

> D <- diag(eigen(R)$values)
> D
```

```
          [,1]      [,2]      [,3]
[1,] 2.433423 0.0000000 0.0000000
[2,] 0.000000 0.4966451 0.0000000
[3,] 0.000000 0.0000000 0.0699315

> GAMMA <- eigen(R)$vectors
> dimnames(GAMMA) <- list(c("w1", "w2", "w3"), c("PC1", "PC2",
+     "PC3"))
> GAMMA

         PC1         PC2        PC3
w1 0.5538345  0.69330367  0.4610828
w2 0.6272670  0.01674325 -0.7786242
w3 0.5475431 -0.72045103  0.4256136

> res$rotation

         PC1         PC2        PC3
w1 0.5538345  0.69330367 -0.4610828
w2 0.6272670  0.01674325  0.7786242
w3 0.5475431 -0.72045103 -0.4256136

> center <- apply(W, MARGIN = 2, FUN = mean)
> center

    w1     w2     w3
4.9000 6.0125 6.2450

> res$center

    w1     w2     w3
4.9000 6.0125 6.2450

> scale <- apply(W, MARGIN = 2, FUN = sigma)
> scale

      w1       w2       w3
2.412986 2.352359 2.195831

> res$scale

      w1       w2       w3
2.579590 2.514778 2.347442

> x <- Z %*% GAMMA
> colnames(x) <- c("PC1", "PC2", "PC3")
> x

             PC1        PC2         PC3
[1,] -3.14634887  0.4336252 -0.06762271
[2,] -1.36030541 -0.5182267  0.22720540
[3,]  0.11795463 -0.5138377  0.29464294
[4,]  0.70034175  0.5688735 -0.18324303
[5,]  1.03272818  1.3443019  0.10120515
[6,] -0.06775909 -0.5075229 -0.41661255
[7,]  1.36832636 -0.5480985 -0.23090583
[8,]  1.35506245 -0.2591149  0.27533061

> res$x
```

```
             PC1         PC2          PC3
[1,] -3.14634887  0.4336252   0.06762271
[2,] -1.36030541 -0.5182267  -0.22720540
[3,]  0.11795463 -0.5138377  -0.29464294
[4,]  0.70034175  0.5688735   0.18324303
[5,]  1.03272818  1.3443019  -0.10120515
[6,] -0.06775909 -0.5075229   0.41661255
[7,]  1.36832636 -0.5480985   0.23090583
[8,]  1.35506245 -0.2591149  -0.27533061
```

summary()

- **Package:** base
- **Input:**

  `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 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`

  $$s_{x_j} \qquad \frac{\lambda_{(k-j+1)}}{k} \qquad \frac{1}{k}\sum_{j=1}^{l} \lambda_{(k-j+1)} \qquad \forall j, l = 1, 2, \ldots, k$$
- **Examples:**

```
> w1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> w2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> w3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> W <- cbind(w1, w2, w3)
> W

      w1  w2   w3
[1,] 1.1 1.2 1.40
[2,] 2.3 3.4 5.60
[3,] 4.5 5.6 7.56
[4,] 6.7 7.5 6.00
[5,] 8.9 7.5 5.40
[6,] 3.4 6.7 6.60
[7,] 5.6 8.6 8.70
[8,] 6.7 7.6 8.70
```

```
> res <- summary(object = prcomp(W, scale. = TRUE))
> n <- 8
> k <- 3
> Z <- scale(W, scale = TRUE)
> colnames(Z) <- c("z1", "z2", "z3")
> Z

             z1         z2         z3
[1,] -1.4731022 -1.9136880 -2.0639484
[2,] -1.0079120 -1.0388592 -0.2747671
[3,] -0.1550634 -0.1640304  0.5601841
[4,]  0.6977852  0.5915036 -0.1043689
[5,]  1.5506339  0.5915036 -0.3599662
[6,] -0.5814877  0.2733840  0.1512284
[7,]  0.2713609  1.0289180  1.0458191
[8,]  0.6977852  0.6312685  1.0458191
attr(,"scaled:center")
    w1     w2     w3
4.9000 6.0125 6.2450
attr(,"scaled:scale")
      w1       w2       w3
2.579590 2.514778 2.347442

> R <- (1/(n - 1)) * t(Z) %*% Z
> dimnames(R) <- list(NULL, NULL)
> R

          [,1]      [,2]      [,3]
[1,] 1.0000000 0.8260355 0.5035850
[2,] 0.8260355 1.0000000 0.8066075
[3,] 0.5035850 0.8066075 1.0000000

> sdev <- sqrt(eigen(R)$values)
> sdev

[1] 1.5599434 0.7047305 0.2644457

> res$sdev

[1] 1.5599434 0.7047305 0.2644457

> GAMMA <- eigen(R)$vectors
> dimnames(GAMMA) <- list(c("w1", "w2", "w3"), c("PC1", "PC2",
+     "PC3"))
> GAMMA

         PC1         PC2        PC3
w1 0.5538345  0.69330367  0.4610828
w2 0.6272670  0.01674325 -0.7786242
w3 0.5475431 -0.72045103  0.4256136

> res$rotation

         PC1         PC2        PC3
w1 0.5538345  0.69330367 -0.4610828
w2 0.6272670  0.01674325  0.7786242
w3 0.5475431 -0.72045103 -0.4256136

> center <- apply(W, MARGIN = 2, FUN = mean)
> center
```

```
    w1     w2     w3
4.9000 6.0125 6.2450

> res$center

    w1     w2     w3
4.9000 6.0125 6.2450

> scale <- apply(W, MARGIN = 2, FUN = sd)
> scale

      w1       w2       w3
2.579590 2.514778 2.347442

> res$scale

      w1       w2       w3
2.579590 2.514778 2.347442

> D <- diag(eigen(S)$values)
> D

         [,1]     [,2]      [,3]
[1,] 15.06232 0.000000 0.0000000
[2,]  0.00000 2.991621 0.0000000
[3,]  0.00000 0.000000 0.4349395

> x <- Z %*% GAMMA
> colnames(x) <- c("PC1", "PC2", "PC3")
> x

             PC1        PC2         PC3
[1,] -3.14634887  0.4336252 -0.06762271
[2,] -1.36030541 -0.5182267  0.22720540
[3,]  0.11795463 -0.5138377  0.29464294
[4,]  0.70034175  0.5688735 -0.18324303
[5,]  1.03272818  1.3443019  0.10120515
[6,] -0.06775909 -0.5075229 -0.41661255
[7,]  1.36832636 -0.5480985 -0.23090583
[8,]  1.35506245 -0.2591149  0.27533061

> res$x

             PC1        PC2         PC3
[1,] -3.14634887  0.4336252  0.06762271
[2,] -1.36030541 -0.5182267 -0.22720540
[3,]  0.11795463 -0.5138377 -0.29464294
[4,]  0.70034175  0.5688735  0.18324303
[5,]  1.03272818  1.3443019 -0.10120515
[6,] -0.06775909 -0.5075229  0.41661255
[7,]  1.36832636 -0.5480985  0.23090583
[8,]  1.35506245 -0.2591149 -0.27533061

> lambda <- sdev^2
> importance <- rbind(sdev, lambda/k, cumsum(lambda)/k)
> dimnames(importance) <- list(c("Standard deviation", "Proportion of Variance",
+     "Cumulative Proportion"), c("PC1", "PC2", "PC3"))
> importance
```

```
                            PC1       PC2       PC3
Standard deviation     1.5599434 0.7047305 0.2644457
Proportion of Variance 0.8111411 0.1655484 0.0233105
Cumulative Proportion  0.8111411 0.9766895 1.0000000
```

```
> res$importance
```

```
                           PC1       PC2       PC3
Standard deviation     1.559943 0.7047305 0.2644457
Proportion of Variance 0.811140 0.1655500 0.0233100
Cumulative Proportion  0.811140 0.9766900 1.0000000
```

Capitolo 5

Analisi dei Gruppi

5.1 Indici di distanza

dist()

- **Package:** stats
- **Input:**

  `x` matrice di dimensione $n \times k$ le cui righe corrispondono ai vettori numerici $x_1, x_2, \ldots, x_n$

  `method = "euclidean" / "maximum" / "manhattan" / "canberra" / "binary" / "minkowski"` indice di distanza

  `p` valore p di potenza per la distanza di *Minkowski*

  `upper = TRUE`

  `diag = TRUE`

- **Description:** matrice di distanza o di dissimilarità per gli n vettori di dimensione $n \times n$
- **Formula:**

`method = "euclidean"`

$$d_{x_i x_j} = \left(\sum_{h=1}^{k} (x_{ih} - x_{jh})^2 \right)^{1/2} \quad \forall\, i, j = 1, 2, \ldots, n$$

`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"`

$$d_{x_i x_j} = \left(\sum_{h=1}^{k} |x_{ih} - x_{jh}|^p \right)^{1/p} \quad \forall\, i, j = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- matrix(data = rnorm(n = 30), nrow = 10, ncol = 3, byrow = FALSE)
> k <- 3
> n <- 10
> dist(x, method = "euclidean", upper = TRUE, diag = TRUE)

           1         2         3         4         5         6         7
1  0.0000000 1.5948359 1.6080407 1.5836525 2.2113048 3.0581815 2.3820407
2  1.5948359 0.0000000 1.4765220 1.5084132 0.9847730 2.9608231 0.8150047
3  1.6080407 1.4765220 0.0000000 1.8622265 2.3977451 1.7540114 1.9745533
4  1.5836525 1.5084132 1.8622265 0.0000000 1.6478362 2.6834204 2.1774463
5  2.2113048 0.9847730 2.3977451 1.6478362 0.0000000 3.6618122 1.0875239
6  3.0581815 2.9608231 1.7540114 2.6834204 3.6618122 0.0000000 3.3142664
7  2.3820407 0.8150047 1.9745533 2.1774463 1.0875239 3.3142664 0.0000000
8  3.4274432 2.2298585 2.1613885 3.3445427 2.8214454 2.8972571 1.7918570
9  1.2371199 2.3024300 2.7601394 1.8380083 2.4297830 4.0248341 3.0452671
10 3.6159883 2.4770211 2.3594738 2.7396964 2.7641401 2.1990887 2.2918994
           8         9        10
1  3.4274432 1.2371199 3.6159883
2  2.2298585 2.3024300 2.4770211
3  2.1613885 2.7601394 2.3594738
4  3.3445427 1.8380083 2.7396964
5  2.8214454 2.4297830 2.7641401
6  2.8972571 4.0248341 2.1990887
7  1.7918570 3.0452671 2.2918994
8  0.0000000 4.4430280 1.8632088
9  4.4430280 0.0000000 4.4151604
10 1.8632088 4.4151604 0.0000000

> dist(x, method = "minkowski", p = 1, upper = TRUE, diag = TRUE)

          1        2        3        4        5        6        7        8
1  0.000000 2.511879 2.548073 2.084588 3.795046 5.216133 3.593517 4.051206
2  2.511879 0.000000 1.680889 2.443684 1.416056 3.923327 1.081638 3.134763
3  2.548073 1.680889 0.000000 3.218951 2.964057 2.668059 2.762527 2.681157
4  2.084588 2.443684 3.218951 0.000000 2.707806 3.603471 3.501799 4.819033
5  3.795046 1.416056 2.964057 2.707806 0.000000 4.320338 1.832726 4.550819
6  5.216133 3.923327 2.668059 3.603471 4.320338 0.000000 4.704210 4.925776
7  3.593517 1.081638 2.762527 3.501799 1.832726 4.704210 0.000000 2.718093
8  4.051206 3.134763 2.681157 4.819033 4.550819 4.925776 2.718093 0.000000
9  1.984456 2.705089 3.960357 3.037213 3.622008 6.628417 3.420478 5.463490
10 5.547416 4.254610 3.611224 3.922487 4.651621 3.572303 3.814418 2.523997
          9       10
1  1.984456 5.547416
2  2.705089 4.254610
3  3.960357 3.611224
4  3.037213 3.922487
5  3.622008 4.651621
6  6.628417 3.572303
7  3.420478 3.814418
8  5.463490 2.523997
9  0.000000 6.959700
10 6.959700 0.000000
```

- **Note 1:** Possiamo ottenere le variabili standardizzate se applichiamo il comando scale() alla matrice `x`.
- **Note 2:** La distanza di dissimilarità calcolata con `method = "binary"` corrisponde al complemento ad uno dell'indice di *Jaccard*.

as.dist()

- **Package:** stats
- **Input:**

  m matrice simmetrica con elementi nulli sulla diagonale di dimensione $n \times n$

  upper = TRUE / FALSE matrice triangolare superiore

  diag = TRUE / FALSE elementi nulli sulla diagonale

- **Description:** oggetto di tipo dist()
- **Examples:**

```
> m <- matrix(data = c(0, 1, 5, 1, 0, 3, 5, 3, 0), nrow = 3, ncol = 3,
+     byrow = TRUE)
> m

     [,1] [,2] [,3]
[1,]    0    1    5
[2,]    1    0    3
[3,]    5    3    0

> n <- 3
> as.dist(m, upper = TRUE, diag = TRUE)

  1 2 3
1 0 1 5
2 1 0 3
3 5 3 0

> as.dist(m, upper = TRUE, diag = FALSE)

  1 2 3
1   1 5
2 1   3
3 5 3

> as.dist(m, upper = FALSE, diag = TRUE)

  1 2 3
1 0
2 1 0
3 5 3 0

> as.dist(m, upper = FALSE, diag = FALSE)

  1 2
2 1
3 5 3
```

mahalanobis()

- **Package:** stats
- **Input:**

  x vettore numerico di dimensione k

  center vettore numerico $\bar{x}$ delle medie di dimensione k

  cov matrice S di covarianza di dimensione $k \times k$

- **Description:** quadrato della distanza di *Mahalanobis*
- **Formula:**

$$MD^2 = (x - \bar{x})^T \, S^{-1} \, (x - \bar{x})$$

- **Example 1:**

```
> X <- matrix(data = c(1.1, 1.2, 1.4, 2.3, 3.4, 5.6, 4.5, 5.6,
+     7.56, 6.7, 7.5, 6, 8.9, 7.5, 5.4, 3.4, 6.7, 6.6, 5.6, 8.6,
+     8.7, 6.7, 7.6, 8.7), nrow = 8, ncol = 3, byrow = TRUE)
> X

     [,1] [,2] [,3]
[1,]  1.1  1.2 1.40
[2,]  2.3  3.4 5.60
[3,]  4.5  5.6 7.56
[4,]  6.7  7.5 6.00
[5,]  8.9  7.5 5.40
[6,]  3.4  6.7 6.60
[7,]  5.6  8.6 8.70
[8,]  6.7  7.6 8.70

> k <- 3
> medie <- apply(X, MARGIN = 2, FUN = mean)
> S <- cov(X)
> x <- c(1.2, 3.4, 5.7)
> as.numeric(t(x - medie) %*% solve(S) %*% (x - medie))

[1] 2.487141

> mahalanobis(x, center = medie, cov = S)

[1] 2.487141
```

- **Example 2:**

```
> X <- matrix(data = c(1.1, 3.4, 2.3, 5.6, 4.5, 6.7, 6.7, 6.7,
+     8.9, 8.6), nrow = 5, ncol = 2, byrow = FALSE)
> X

     [,1] [,2]
[1,]  1.1  6.7
[2,]  3.4  6.7
[3,]  2.3  6.7
[4,]  5.6  8.9
[5,]  4.5  8.6

> k <- 2
> medie <- apply(X, MARGIN = 2, FUN = mean)
> S <- cov(X)
> x <- c(1.4, 6.7)
> as.numeric(t(x - medie) %*% solve(S) %*% (x - medie))

[1] 1.530355
```

```
> mahalanobis(x, center = medie, cov = S)

[1] 1.530355
```

- **Example 3:**

```
> X <- matrix(data = c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7,
+     1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6, 1.4, 5.6, 7.56, 6,
+     5.4, 6.6, 8.7, 8.7, 1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6),
+     nrow = 8, ncol = 4, byrow = TRUE)
> X

     [,1] [,2] [,3] [,4]
[1,] 1.10  2.3 4.50  6.7
[2,] 8.90  3.4 5.60  6.7
[3,] 1.20  3.4 5.60  7.5
[4,] 7.50  6.7 8.60  7.6
[5,] 1.40  5.6 7.56  6.0
[6,] 5.40  6.6 8.70  8.7
[7,] 1.50  6.4 9.60  8.8
[8,] 8.86  7.8 8.60  8.6

> k <- 4
> medie <- apply(X, MARGIN = 2, FUN = mean)
> S <- cov(X)
> x <- c(1.1, 2.4, 10.4, 7.8)
> as.numeric(t(x - medie) %*% solve(S) %*% (x - medie))

[1] 114.4839

> mahalanobis(x, center = medie, cov = S)

[1] 114.4839
```

5.2 Criteri di Raggruppamento

hclust()

- **Package:** `stats`
- **Input:**

  `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*

- **Description:** 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

  `labels` 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"`

$$d_{(xy)z} = \frac{n_x\, d_{xz} + n_y\, d_{yz}}{n_{(xy)}} - \frac{n_x\, n_y\, d_{xy}}{n^2_{(xy)}}$$

- **Example 1:**

```
> x <- matrix(data = rnorm(n = 30), nrow = 3, ncol = 10, byrow = FALSE)
> k <- 3
> n <- 10
> d <- dist(x, method = "euclidean", upper = TRUE, diag = TRUE)
> hclust(d = d, method = "single")

Call:
hclust(d = d, method = "single")

Cluster method   : single
Distance         : euclidean
Number of objects: 3

> res <- hclust(d = d, method = "single")
> res$merge

     [,1] [,2]
[1,]   -2   -3
[2,]   -1    1

> res$height

[1] 2.985362 3.761878
```

```
> res$order

[1] 1 2 3

> res$labels

NULL

> res$method

[1] "single"

> res$dist.method

[1] "euclidean"
```

- **Example 2:**

```
> x <- matrix(data = rnorm(n = 100), nrow = 20, ncol = 5, byrow = FALSE)
> k <- 3
> n <- 10
> d <- dist(x, method = "euclidean", upper = TRUE, diag = TRUE)
> hclust(d = d, method = "median")

Call:
hclust(d = d, method = "median")

Cluster method   : median
Distance         : euclidean
Number of objects: 20

> res <- hclust(d = d, method = "median")
> res$merge

      [,1] [,2]
 [1,]   -6  -16
 [2,]   -2    1
 [3,]  -14    2
 [4,]  -12  -20
 [5,]  -19    4
 [6,]    3    5
 [7,]  -15    6
 [8,]  -13  -18
 [9,]  -10    8
[10,]  -11    9
[11,]    7   10
[12,]   -4  -17
[13,]   11   12
[14,]   -5   13
[15,]   -7   14
[16,]   -1   -8
[17,]   15   16
[18,]   -3   17
[19,]   -9   18

> res$height

 [1] 1.129097 1.070475 1.196478 1.351082 1.274444 1.390697 1.335846 1.440786
 [9] 1.606760 1.559425 1.650469 1.819976 1.762757 1.643485 2.162323 2.422278
[17] 2.680234 2.464257 2.140949
```

```
> res$order
```

```
 [1]  9  3  7  5 15 14  2  6 16 19 12 20 11 10 13 18  4 17  1  8
```

```
> res$labels
```

```
NULL
```

```
> res$method
```

```
[1] "median"
```

```
> res$dist.method
```

```
[1] "euclidean"
```

kmeans()

- **Package:** stats
- **Input:**

  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 concesse al criterio di ottimizzazione
- **Description:** analisi di ragguppamento non gerarchica con il metodo *k-means*
- **Output:**

  cluster gruppo di appartenenza di ciascuna osservazione

  centers centroidi dei gruppi ottenuti

  withinss devianza di ciascun gruppo

  size numero di osservazioni in ciascun gruppo
- **Example 1:**

```
> x <- matrix(data = rnorm(n = 100, mean = 0, sd = 0.3), nrow = 50,
+     ncol = 2, byrow = FALSE)
> kmeans(x, centers = 2, iter.max = 10)

K-means clustering with 2 clusters of sizes 29, 21

Cluster means:
         [,1]       [,2]
1 -0.05916688 -0.1945814
2  0.04105267  0.2989030

Clustering vector:
 [1] 1 2 2 1 1 2 2 1 1 2 2 1 1 1 1 1 1 1 1 2 1 2 2 1 1 1 1 2 2 2 1 2 1 2 1 2 1 2
[39] 2 1 1 1 2 2 1 1 1 2 2 1

Within cluster sum of squares by cluster:
[1] 2.771814 2.263145

Available components:
[1] "cluster"  "centers"  "withinss" "size"

> res <- kmeans(x, centers = 2, iter.max = 10)
> res$cluster
```

```
 [1] 1 2 1 1 1 2 2 1 1 2 2 1 1 1 1 1 1 1 1 2 1 2 2 1 2 2 1 1 2 2 1 2 1 2 1 2 2 2
[39] 2 2 2 2 2 2 1 2 1 2 1 2

> res$centers

         [,1]       [,2]
1  0.07741224 -0.2356923
2 -0.10429336  0.2419507

> res$withinss

[1] 2.079959 2.784218

> res$size

[1] 24 26
```

- **Example 2:**

```
> x <- matrix(data = rnorm(n = 80, mean = 0, sd = 0.3), nrow = 40,
+     ncol = 2, byrow = FALSE)
> kmeans(x, centers = 5, iter.max = 15)

K-means clustering with 5 clusters of sizes 5, 5, 7, 13, 10

Cluster means:
        [,1]        [,2]
1 -0.2826432  0.37367857
2 -0.4721982 -0.53828582
3  0.2601737  0.14589161
4 -0.2726225 -0.07709169
5  0.2381249 -0.14376129

Clustering vector:
 [1] 4 4 3 4 5 5 5 4 5 1 1 4 4 3 2 1 4 2 2 4 5 3 1 4 4 5 4 3 4 5 3 1 3 5 2 5 3 5
[39] 2 4

Within cluster sum of squares by cluster:
[1] 0.2127299 0.2585805 0.1444599 0.4426205 0.2739510

Available components:
[1] "cluster"  "centers"  "withinss" "size"

> res <- kmeans(x, centers = 5, iter.max = 15)
> res$cluster

 [1] 2 3 5 3 5 5 2 3 2 1 1 3 3 5 4 1 2 4 4 3 2 5 1 3 3 2 3 5 3 5 5 1 5 5 4 5 2 2
[39] 4 3

> res$centers

         [,1]        [,2]
1 -0.28264316  0.37367857
2  0.06019474 -0.09067425
3 -0.30619549 -0.08337684
4 -0.47219821 -0.53828582
5  0.32226949  0.02036143

> res$withinss

[1] 0.2127299 0.2084292 0.3159412 0.2585805 0.4271144
```

```
> res$size
[1]  5  8 11  5 11
```

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()`
- **Input:**

  `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`

$$\mu_0$$

- **Example 1:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1, 6.7, 7.6, 6.8)
> xmedio <- 7.018182
> sigmax <- 1.2
> n <- 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] 0.1520925

> 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 7.727324

> 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"
```

- **Example 2:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> xmedio <- 4.68
> sigmax <- 1.45
> n <- 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()`
- **Input:**

  `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	$P(t_{df} \le t)$	$1 - P(t_{df} \le t)$	$2\,P(t_{df} \le -\mid t\mid)$

  `conf.int`

  $$\bar{x} \mp t_{1-\alpha\,/\,2,\,df}\; s_x \,/\, \sqrt{n}$$

  `estimate`

  $$\bar{x}$$

  `null.value`

  $$\mu_0$$

- **Example 1:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1, 6.7, 7.6, 6.8)
> xmedio <- 7.018182
> sx <- 0.4643666
> n <- 11
> mu0 <- 6.5
> t <- (xmedio - mu0)/(sx/sqrt(n))
> t

[1] 3.700988

> 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
10

> p.value <- 2 * pt(-abs(t), df = n - 1)
> p.value

[1] 0.004101807

> 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"
```

- **Example 2:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> xmedio <- 4.68
> sx <- 3.206556
> n <- 5
> mu0 <- 5.2
> t <- (xmedio - mu0)/(sx/sqrt(n))
> t

[1] -0.3626181

> 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
 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] 0.6985349 8.6614651

> 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()
- **Input:**

  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	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\mid z \mid)$

  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

$$(\mu_x - \mu_y)_{|H_0}$$

- **Example 1:**

```
> x <- c(154, 109, 137, 115, 140)
> xmedio <- 131
> sigmax <- 15.5
> nx <- 5
> y <- c(108, 115, 126, 92, 146)
> 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 <- 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"
```

- **Example 2:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1, 6.7, 7.6, 6.8)
> xmedio <- 7.018182
> sigmax <- 0.5
> nx <- 11
> y <- c(4.5, 5.4, 6.1, 6.1, 5.4, 5, 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)
> 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] 0.0829646

> 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 e supposte uguali

- **Package:** `stats`
- **Sintassi:** `t.test()`
- **Input:**

  `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 $1 - \alpha$

  `var.equal = TRUE`

- **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	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\,P(t_{df} \leq -\,|\,t\,|)$

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

$$(\,\mu_x - \mu_y\,)_{|\,H_0}$$

- **Example 1:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1, 6.7, 7.6, 6.8)
> xmedio <- 7.018182
> sx <- 0.4643666
> nx <- 11
> y <- c(4.5, 5.4, 6.1, 6.1, 5.4, 5, 4.1, 5.5)
> ymedio <- 5.2625
> sy <- 0.7069805
> ny <- 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 = TRUE)
> res$statistic

       t
2.073455

> parameter <- nx + ny - 2
> parameter

[1] 17

> res$parameter

df
17

> p.value <- 2 * pt(-abs(t), df = nx + ny - 2)
> p.value

[1] 0.05364035

> 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] 1.190256 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"
```

- **Example 2:**

```
> x <- c(154, 109, 137, 115, 140)
> xmedio <- 131
> sx <- 18.61451
> nx <- 5
> y <- c(108, 115, 126, 92, 146)
> ymedio <- 117.4
> sy <- 20.19406
> ny <- 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] 0.2930997
```

```
> res <- t.test(x, y, mu = 10, alternative = "two.sided", conf.level = 0.95,
+     var.equal = TRUE)
> res$statistic

        t
0.2930998

> parameter <- nx + ny - 2
> parameter

[1] 8

> res$parameter

df
 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 supposte diverse

- **Package:** stats
- **Sintassi:** `t.test()`
- **Input:**

  `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`

$$df = \frac{\left(s_x^2\,/\,n_x + s_y^2\,/\,n_y\right)^2}{s_x^4\,/\,(n_x^2\,(n_x-1)) + s_y^4\,/\,(n_y^2\,(n_y-1))} = \left(\frac{1}{n_x-1}\,C^2 + \frac{1}{n_y-1}\,(1-C)^2\right)^{-1}$$

$$\text{dove} \quad C = \frac{s_x^2\,/\,n_x}{s_x^2\,/\,n_x + s_y^2\,/\,n_y}$$

  `p.value`

alternative	less	greater	two.sided
p.value	$P(t_{df} \le t)$	$1 - P(t_{df} \le t)$	$2\,P(t_{df} \le -\,\vert\,t\,\vert)$

  `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`

$$(\,\mu_x - \mu_y\,)_{|\,H_0}$$

- **Example 1:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1, 6.7, 7.6, 6.8)
> xmedio <- 7.018182
> sx <- 0.4643666
> nx <- 11
> y <- c(4.5, 5.4, 6.1, 6.1, 5.4, 5, 4.1, 5.5)
> ymedio <- 5.2625
> sy <- 0.7069805
> ny <- 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)
> 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

> C <- (sx^2/nx)/(sx^2/nx + sy^2/ny)
> gl <- as.numeric(solve(solve(nx - 1) * C^2 + solve(ny - 1) *
+     (1 - C)^2))
> gl

[1] 11.30292

> res$parameter

      df
11.30292

> p.value <- 2 * pt(-abs(t), df = gl)
> p.value

[1] 0.0777921

> 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 2.384204

> 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"
```

- **Example 2:**

```
> x <- c(154, 109, 137, 115, 140)
> xmedio <- 131
> sx <- 18.61451
> nx <- 5
> y <- c(108, 115, 126, 92, 146)
> ymedio <- 117.4
> sy <- 20.19406
> ny <- 5
> mu0 <- 10
> t <- (xmedio - ymedio - mu0)/sqrt(sx^2/nx + sy^2/ny)
> t

[1] 0.2930997

> res <- t.test(x, y, mu = 10, alternative = "two.sided", conf.level = 0.95)
> 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] 7.947511

> C <- (sx^2/nx)/(sx^2/nx + sy^2/ny)
> gl <- as.numeric(solve(solve(nx - 1) * C^2 + solve(ny - 1) *
+     (1 - C)^2))
> gl

[1] 7.947511

> res$parameter

      df
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()`
- **Input:**

  `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 $1 - \alpha$

  `paired = TRUE`

- **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`

$$t = \frac{(\bar{x}-\bar{y}) - (\mu_x - \mu_y)_{|H_0}}{s_{x-y} / \sqrt{n}}$$

$$\text{dove} \quad s_{x-y}^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left((x_i - y_i) - (\bar{x}-\bar{y})\right)^2 = s_x^2 + s_y^2 - 2\, s_{xy}$$

`parameter`

$$df = n-1$$

`p.value`

alternative	less	greater	two.sided
p.value	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\,P(t_{df} \leq -\,|\,t\,|)$

`conf.int`

$$\bar{x}-\bar{y} \mp t_{1-\alpha\,/\,2,\,df}\; s_{x-y} / \sqrt{n}$$

`estimate`

$$\bar{x}-\bar{y}$$

`null.value`

$$(\mu_x - \mu_y)_{|H_0}$$

- **Example 1:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1)
> xmedio <- 7.0125
> y <- c(4.5, 5.4, 6.1, 6.1, 5.4, 5, 4.1, 5.5)
> ymedio <- 5.2625
> n <- 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 = TRUE)
> res$statistic

       t
1.815412

> parameter <- n - 1
> parameter

[1] 7

> res$parameter

df
 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"
```

- **Example 2:**

```
> x <- c(154, 109, 137, 115, 140)
> xmedio <- 131
> y <- c(108, 115, 126, 92, 146)
> ymedio <- 117.4
> n <- 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 = TRUE)
> res$statistic

        t
0.3680758

> parameter <- n - 1
> parameter

[1] 4

> res$parameter

df
 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"
```

Test di Fisher con k campioni indipendenti

- **Package:** stats
- **Sintassi:** oneway.test()
- **Input:**

  formula modello di regressione lineare con una variabile esplicativa fattore f a k livelli ed n unità

  var.equal = TRUE
- **Output:**

  statistic valore empirico della statistica F

  parameter gradi di libertà

  p.value p-value
- **Formula:**

  statistic

$$Fvalue = \frac{\left[\sum_{j=1}^{k} n_j \, (\bar{y}_j - \bar{y})^2\right] / \,(k-1)}{\left[\sum_{j=1}^{k} \sum_{i=1}^{n_j} (y_{ij} - \bar{y}_j)^2\right] / \,(n-k)}$$

  parameter

f	$k-1$
Residuals	$n-k$

  p.value

$$P(F_{k-1,\,n-k} \geq Fvalue)$$

- **Examples:**

```
> y <- c(1, 4, 10, 2.1, 3.5, 5.6, 8.4, 12, 16.5, 22, 1.2, 3.4)
> f <- factor(rep(letters[1:4], each = 3))
> f

 [1] a a a b b b c c c d d d
Levels: a b c d

> n <- 12
> k <- 4
> oneway.test(formula = y ~ f, var.equal = TRUE)

        One-way analysis of means

data:  y and f
F = 1.0597, num df = 3, denom df = 8, p-value = 0.4184
```

6.2 Test di ipotesi sulla media con uno o due campioni (summarized data)

Test Z con un campione

- **Package:** BSDA
- **Sintassi:** zsum.test()
- **Input:**

  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	$\Phi(z)$	$1-\Phi(z)$	$2\,\Phi(-\,|\,z\,|)$

`conf.int`

$$\bar{x} \mp z_{1-\alpha\,/\,2}\,\sigma_x \,/\, \sqrt{n}$$

`estimate`

$$\bar{x}$$

`null.value`

$$\mu_0$$

- **Example 1:**

```
> 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] 0.1520925

> res$p.value

[1] 0.1520925
```

```
> 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 7.727324

> res$conf.int

[1] 6.309040 7.727324
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"
```

- **Example 2:**

```
> 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()`
- **Input:**

  `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	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\,P(t_{df} \leq -\mid t \mid)$

`conf.int`

$$\bar{x} \mp t_{1-\alpha\,/\,2,\,df}\; s_x \,/\, \sqrt{n}$$

`estimate`

$$\bar{x}$$

`null.value`

$$\mu_0$$

- **Example 1:**

```
> 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 = 0.95)
> res$statistic

       t
1.432179

> parameter <- n - 1
> parameter

[1] 10

> res$parameter

df
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"
```

- **Example 2:**

```
> 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 = 0.95)
> res$statistic

         t
-0.8019002

> parameter <- n - 1
> parameter

[1] 4
```

```
> res$parameter

df
 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()`
- **Input:**

  `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`

$$(\mu_x - \mu_y)_{|H_0}$$

- **Example 1:**

```
> 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"
```

- **Example 2:**

```
> 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 e supposte uguali

- **Package:** BSDA
- **Sintassi:** `tsum.test()`
- **Input:**

  `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 $1 - \alpha$

  `var.equal = TRUE`

- **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	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\,P(t_{df} \leq -\mid t \mid)$

  `conf.int`

$$\bar{x} - \bar{y} \mp t_{1-\alpha/2,\, df}\, s_P \sqrt{1 / n_x + 1 / n_y}$$

estimate
$\bar{x} \quad \bar{y}$

null.value
$(\mu_x - \mu_y)_{|H_0}$

- **Example 1:**

```
> 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))
> 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
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"
```

- **Example 2:**

```
> 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 = TRUE)
> res$statistic

        t
0.3916284

> parameter <- nx + ny - 2
> parameter

[1] 8

> res$parameter

df
 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 supposte diverse

- **Package:** BSDA
- **Sintassi:** tsum.test()
- **Input:**

  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 $1 - \alpha$

  var.equal = FALSE

- **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

$$df = \frac{\left(s_x^2 / n_x + s_y^2 / n_y\right)^2}{s_x^4 / (n_x^2 (n_x - 1)) + s_y^4 / (n_y^2 (n_y - 1))} = \left(\frac{1}{n_x - 1} C^2 + \frac{1}{n_y - 1} (1 - C)^2\right)^{-1}$$

$$\text{dove} \quad C = \frac{s_x^2 / n_x}{s_x^2 / n_x + s_y^2 / n_y}$$

  p.value

alternative	less	greater	two.sided
p.value	$P(t_{df} \le t)$	$1 - P(t_{df} \le t)$	$2\,P(t_{df} \le -\lvert t \rvert)$

  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

$$(\mu_x - \mu_y)_{|H_0}$$

- **Example 1:**

```
> 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 = FALSE)
> 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

> C <- (sx^2/nx)/(sx^2/nx + sy^2/ny)
> gl <- as.numeric(solve(solve(nx - 1) * C^2 + solve(ny - 1) *
+     (1 - C)^2))
> gl

[1] 10.92501

> res$parameter

      df
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"
```

- **Example 2:**

```
> 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 = FALSE)
> 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

> C <- (sx^2/nx)/(sx^2/nx + sy^2/ny)
> gl <- as.numeric(solve(solve(nx - 1) * C^2 + solve(ny - 1) *
+     (1 - C)^2))
> gl

[1] 7.852026

> res$parameter
```

```
      df
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()
- **Input:**

  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	$P(\chi^2_{df} \le c)$	$P(\chi^2_{df} \ge c)$	$2 \min\left(P(\chi^2_{df} \le c),\, P(\chi^2_{df} \ge c)\right)$

  conf.int

  $$\frac{(n-1)\, s_x^2}{\chi^2_{1-\alpha\,/\,2,\, df}} \qquad \frac{(n-1)\, s_x^2}{\chi^2_{\alpha\,/\,2,\, df}}$$

  estimate

  $$s_x^2$$

  null.value

  $$\sigma_0^2$$

- **Example 1:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1, 6.7, 7.6, 6.8)
> sx <- 0.4643666
> n <- 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] 0.1357228

> 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] 0.1052748 0.6641150

> res$conf.int

[1] 0.1052749 0.6641151
attr(,"conf.level")
[1] 0.95

> sx^2

[1] 0.2156363

> res$estimate

 var of x
0.2156364

> var0

[1] 0.5

> res$null.value

variance
     0.5

> res$alternative

[1] "two.sided"
```

- **Example 2:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> sx <- 3.206556
> n <- 5
> var0 <- 12
> c <- (n - 1) * sx^2/var0
> c

[1] 3.427334

> 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] 0.9780261

> 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]  3.690833 84.901796

> res$conf.int

[1]  3.690832 84.901785
attr(,"conf.level")
[1] 0.95

> sx^2

[1] 10.28200

> 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()
- **Input:**

  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	$P(F_{df_1,df_2} \leq Fval)$	$P(F_{df_1,df_2} \geq Fval)$	$2\,\min\left(P(F_{df_1,df_2} \leq Fval),\, P(F_{df_1,df_2} \geq Fval)\right)$

  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 $\frac{\sigma_x^2}{\sigma_y^2}\Big| H_0$

- **Example 1:**

```
> x <- c(7, -4, 18, 17, -3, -5, 1, 10, 11, -2, -3)
> nx <- 11
> y <- c(-1, 12, -1, -3, 3, -5, 5, 2, -11, -1, -3)
> ny <- 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"
```

- **Example 2:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1, 6.7, 7.6, 6.8)
> nx <- 11
> y <- c(4.5, 5.4, 6.1, 6.1, 5.4, 5, 4.1, 5.5)
> ny <- 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()`
- **Input:**

  `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 $1 - \alpha$

  `correct = FALSE`

- **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	$\Phi(z)$	$1 - \Phi(z)$	$P(\chi^2_1 \geq z^2)$

  `conf.int`

$$\frac{\left(2\,x + z^2_{1-\alpha\,/\,2}\right) \mp \sqrt{\left(2\,x + z^2_{1-\alpha\,/\,2}\right)^2 - 4\,\left(n + z^2_{1-\alpha\,/\,2}\right)\,x^2\,/\,n}}{2\,\left(n + z^2_{1-\alpha\,/\,2}\right)}$$

  `estimate`

$$\frac{x}{n}$$

  `null.value`

$$p_0$$

- **Example 1:**

```
> 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 = FALSE)
> 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 <- ((2 * x + zc^2) - sqrt((2 * x + zc^2)^2 - 4 * (n + zc^2) *
+     x^2/n))/(2 * (n + zc^2))
> upper <- ((2 * x + zc^2) + sqrt((2 * x + zc^2)^2 - 4 * (n + zc^2) *
+     x^2/n))/(2 * (n + zc^2))
> 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"
```

- **Example 2:**

```
> 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 = FALSE)
> 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()`
- **Input:**

  `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`

$$1 - \beta = \Phi \left((\gamma \, / \, \xi) \, \sqrt{n} - (\delta \, / \, \xi) \, \Phi^{-1}(1 - \alpha \, / \, 2) \right)$$

- **Example 1:**

```
> n <- 23
> p1 <- 0.23
> p2 <- 0.31
> power.prop.test(n, p1, p2, sig.level = NULL, power = 0.9, alternative = "one.sided")

     Two-sample comparison of proportions power calculation

              n = 23
             p1 = 0.23
             p2 = 0.31
      sig.level = 0.7470593
          power = 0.9
    alternative = one.sided

 NOTE: n is number in *each* group
```

- **Example 2:**

```
> p1 <- 0.23
> p2 <- 0.31
> power.prop.test(n = NULL, p1, p2, sig.level = 0.05, power = 0.9,
+     alternative = "one.sided")

     Two-sample comparison of proportions power calculation

              n = 525.6022
             p1 = 0.23
             p2 = 0.31
      sig.level = 0.05
          power = 0.9
    alternative = one.sided

 NOTE: n is number in *each* group
```

- **Example 3:**

```
> n <- 23
> p1 <- 0.23
> p2 <- 0.31
> power.prop.test(n, p1, p2, sig.level = 0.05, power = NULL, alternative = "one.sided")

     Two-sample comparison of proportions power calculation

              n = 23
             p1 = 0.23
             p2 = 0.31
      sig.level = 0.05
          power = 0.1496353
    alternative = one.sided

 NOTE: n is number in *each* group
```

Test con due campioni indipendenti

- **Package:** `stats`
- **Sintassi:** `prop.test()`
- **Input:**

  `x` numero di successi nei due campioni

  `n` dimensione dei due campioni

`alternative = "less" / "greater" / "two.sided"` ipotesi alternativa

`conf.level` livello di confidenza $1 - \alpha$

`correct = FALSE`

- **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`

`correct = TRUE`

$$z^2 = \left(\frac{\left| \frac{x_1}{n_1} - \frac{x_2}{n_2} \right| - 0.5 \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}{\sqrt{\frac{x_1+x_2}{n_1+n_2} \left(1 - \frac{x_1+x_2}{n_1+n_2}\right) \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}} \right)^2$$

`correct = FALSE`

$$z^2 = \left(\frac{\frac{x_1}{n_1} - \frac{x_2}{n_2}}{\sqrt{\frac{x_1+x_2}{n_1+n_2} \left(1 - \frac{x_1+x_2}{n_1+n_2}\right) \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}} \right)^2$$

`parameter`

$$1$$

`p.value`

alternative	less	greater	two.sided
p.value	$\Phi(z)$	$1 - \Phi(z)$	$1 - P(\chi^2_1 \leq z^2)$

`conf.int`

`correct = TRUE`

$$\left| \frac{x_1}{n_1} - \frac{x_2}{n_2} \right| \mp 0.5 \left(\frac{1}{n_1} + \frac{1}{n_2} \right) \mp z_{1-\alpha\,/\,2} \sqrt{\frac{\frac{x_1}{n_1}\left(1 - \frac{x_1}{n_1}\right)}{n_1} + \frac{\frac{x_2}{n_2}\left(1 - \frac{x_2}{n_2}\right)}{n_2}}$$

`correct = FALSE`

$$\frac{x_1}{n_1} - \frac{x_2}{n_2} \mp z_{1-\alpha\,/\,2} \sqrt{\frac{\frac{x_1}{n_1}\left(1 - \frac{x_1}{n_1}\right)}{n_1} + \frac{\frac{x_2}{n_2}\left(1 - \frac{x_2}{n_2}\right)}{n_2}}$$

`estimate`

$$\frac{x_1}{n_1} \qquad \frac{x_2}{n_2}$$

- **Example 1:**

```
> x <- c(9, 11)
> n <- c(23, 32)
> x1 <- 9
> x2 <- 11
> n1 <- 23
> n2 <- 32
> z <- (x1/n1 - x2/n2)/sqrt((x1 + x2)/(n1 + n2) * (1 - (x1 + x2)/(n1 +
+     n2)) * (1/n1 + 1/n2))
> z^2
```

```
[1] 0.1307745

> res <- prop.test(x = c(9, 11), n = c(23, 32), alternative = "two.sided",
+     conf.level = 0.95, correct = FALSE)
> 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 <- (x1/n1 - x2/n2) - qnorm(1 - 0.05/2) * sqrt(x1/n1 * (1 -
+     x1/n1)/n1 + x2/n2 * (1 - x2/n2)/n2)
> upper <- (x1/n1 - x2/n2) + qnorm(1 - 0.05/2) * sqrt(x1/n1 * (1 -
+     x1/n1)/n1 + x2/n2 * (1 - x2/n2)/n2)
> c(lower, upper)

[1] -0.2110231  0.3061318

> res$conf.int

[1] -0.2110231  0.3061318
attr(,"conf.level")
[1] 0.95

> c(x1/n1, x2/n2)

[1] 0.3913043 0.3437500

> res$estimate

   prop 1    prop 2
0.3913043 0.3437500

> res$alternative

[1] "two.sided"
```

- **Example 2:**

```
> x <- c(4, 11)
> n <- c(20, 24)
> x1 <- 4
> x2 <- 11
> n1 <- 20
> n2 <- 24
> z <- (x1/n1 - x2/n2)/sqrt((x1 + x2)/(n1 + n2) * (1 - (x1 + x2)/(n1 +
+     n2)) * (1/n1 + 1/n2))
> z^2
```

```
[1] 3.240153

> res <- prop.test(x = c(4, 11), n = c(20, 24), alternative = "two.sided",
+     conf.level = 0.95, correct = FALSE)
> 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 <- (x1/n1 - x2/n2) - qnorm(1 - 0.05/2) * sqrt(x1/n1 * (1 -
+     x1/n1)/n1 + x2/n2 * (1 - x2/n2)/n2)
> upper <- (x1/n1 - x2/n2) + qnorm(1 - 0.05/2) * sqrt(x1/n1 * (1 -
+     x1/n1)/n1 + x2/n2 * (1 - x2/n2)/n2)
> c(lower, upper)

[1] -0.523793280  0.007126613

> res$conf.int

[1] -0.523793280  0.007126613
attr(,"conf.level")
[1] 0.95

> c(x1/n1, x2/n2)

[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()
- **Input:**

  `x` numero di successi nei k campioni

  `n` dimensione dei k campioni

  `correct = FALSE`

- **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`

$$c = \sum_{i=1}^{k} \left(\frac{\frac{x_i}{n_i} - \hat{p}}{\sqrt{\hat{p}\,(1 - \hat{p})\,/\,n_i}} \right)^2$$

$$\text{dove} \quad \hat{p} = \frac{\sum_{j=1}^{k} x_j}{\sum_{j=1}^{k} n_j}$$

  `parameter`

$$df = k - 1$$

  `p.value`

$$P(\chi^2_{df} \geq c)$$

  `estimate`

$$\frac{x_i}{n_i} \quad \forall\, i = 1, 2, \ldots, k$$

- **Example 1:**

```
> k <- 3
> x <- c(10, 21, 32)
> n <- c(23, 55, 81)
> phat <- sum(x)/sum(n)
> statistic <- sum(((x/n - phat)/sqrt(phat * (1 - phat)/n))^2)
> statistic

[1] 0.1911084

> prop.test(x, n, correct = FALSE)$statistic

X-squared
0.1911084

> parameter <- k - 1
> parameter

[1] 2

> prop.test(x, n, correct = FALSE)$parameter

df
 2
```

```
> p.value <- 1 - pchisq(statistic, df = k - 1)
> p.value

[1] 0.9088691

> prop.test(x, n, correct = FALSE)$p.value

[1] 0.9088691

> estimate <- x/n
> estimate

[1] 0.4347826 0.3818182 0.3950617

> prop.test(x, n, correct = FALSE)$estimate

   prop 1    prop 2    prop 3
0.4347826 0.3818182 0.3950617
```

- **Example 2:**

```
> k <- 4
> x <- c(17, 14, 21, 34)
> n <- c(26, 22, 33, 45)
> phat <- sum(x)/sum(n)
> statistic <- sum(((x/n - phat)/sqrt(phat * (1 - phat)/n))^2)
> statistic

[1] 1.747228

> prop.test(x, n, correct = FALSE)$statistic

X-squared
 1.747228

> parameter <- k - 1
> parameter

[1] 3

> prop.test(x, n, correct = FALSE)$parameter

df
 3

> p.value <- 1 - pchisq(statistic, df = k - 1)
> p.value

[1] 0.6264855

> prop.test(x, n, correct = FALSE)$p.value

[1] 0.6264855

> estimate <- x/n
> estimate

[1] 0.6538462 0.6363636 0.6363636 0.7555556

> prop.test(x, n, correct = FALSE)$estimate

   prop 1    prop 2    prop 3    prop 4
0.6538462 0.6363636 0.6363636 0.7555556
```

6.5 Test di ipotesi sull'omogeneità delle varianze

Test di Bartlett

- **Package:** stats
- **Sintassi:** bartlett.test()
- **Input:**

  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

$$df \,=\, k-1$$

  p.value

$$P(\chi^2_{df} \,\geq\, c)$$

- **Example 1:**

```
> x <- c(1, 4, 10, 2.1, 3.5, 5.6, 8.4, 12, 16.5, 22, 1.2, 3.4)
> g <- factor(rep(1:4, each = 3))
> g

 [1] 1 1 1 2 2 2 3 3 3 4 4 4
Levels: 1 2 3 4

> n <- 12
> k <- 4
> s2 <- tapply(x, g, var)
> s2

         1          2          3          4
 21.000000   3.103333  16.470000 130.573333

> enne <- tapply(x, g, length)
> enne

1 2 3 4
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
```

- **Example 2:**

```
> x <- c(0.7, -1.6, -0.2, -1.2, -0.1, 3.4, 3.7, 0.8, 0, 2, 1.9,
+     0.8)
> g <- factor(rep(1:2, c(8, 4)))
> g

 [1] 1 1 1 1 1 1 1 1 2 2 2 2
Levels: 1 2

> n <- 12
> k <- 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 Modelli di analisi della varianza

Anova ad un fattore

- **Package:** `stats`
- **Sintassi:** `anova()`
- **Input:**

  `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

f	$k-1$
Residuals	$n-k$

Sum Sq

f	$\sum_{j=1}^{k} n_j \, (\bar{y}_j - \bar{y})^2$
Residuals	$\sum_{j=1}^{k} \sum_{i=1}^{n_j} (y_{ij} - \bar{y}_j)^2$

Mean Sq

f	$\left[\sum_{j=1}^{k} n_j \, (\bar{y}_j - \bar{y})^2\right] / \, (k-1)$
Residuals	$\left[\sum_{j=1}^{k} \sum_{i=1}^{n_j} (y_{ij} - \bar{y}_j)^2\right] / \, (n-k)$

F value

$$Fvalue = \frac{\left[\sum_{j=1}^{k} n_j \, (\bar{y}_j - \bar{y})^2\right] / \, (k-1)}{\left[\sum_{j=1}^{k} \sum_{i=1}^{n_j} (y_{ij} - \bar{y}_j)^2\right] / \, (n-k)}$$

Pr(>F)

$$P(F_{k-1, \, n-k} \geq Fvalue)$$

- **Examples:**

```
> y <- c(1, 4, 10, 2.1, 3.5, 5.6, 8.4, 12, 16.5, 22, 1.2, 3.4)
> f <- factor(rep(letters[1:4], each = 3))
> f

 [1] a a a b b b c c c d d d
Levels: a b c d

> n <- 12
> k <- 4
> modello <- lm(formula = y ~ f)
> anova(modello)

Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value Pr(>F)
f          3 136.03   45.34  1.0597 0.4184
Residuals  8 342.29   42.79

> res <- anova(object = modello)
> res$Df

[1] 3 8

> res$"Sum Sq"

[1] 136.0292 342.2933

> res$"Mean Sq"

[1] 45.34306 42.78667

> res$"F value"
```

```
[1] 1.059747       NA

> res$"Pr(>F)"

[1] 0.4183517       NA
```

Anova a due fattori senza interazione

- **Package:** stats
- **Sintassi:** anova()
- **Input:**

  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

f	$k-1$
g	$h-1$
Residuals	$k\,h\,l-(k+h-1)$

Sum Sq

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	$l\,\sum_{j=1}^{k}\sum_{i=1}^{h}(\bar{y}_{ij\cdot}-\bar{y}_{i\cdot\cdot}-\bar{y}_{\cdot j\cdot}+\bar{y})^2+\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij\cdot})^2$

Mean Sq

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	$\frac{\left[l\,\sum_{j=1}^{k}\sum_{i=1}^{h}(\bar{y}_{ij\cdot}-\bar{y}_{i\cdot\cdot}-\bar{y}_{\cdot j\cdot}+\bar{y})^2+\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij\cdot})^2\right]}{\left[k\,h\,l-(k+h-1)\right]}$

F value

$$F_f value = \frac{\left[h\,l\,\sum_{j=1}^{k}(\bar{y}_{\cdot j \cdot}-\bar{y})^2\right]/(k-1)}{\frac{\left[l\,\sum_{j=1}^{k}\sum_{i=1}^{h}(\bar{y}_{ij\cdot}-\bar{y}_{i\cdot\cdot}-\bar{y}_{\cdot j\cdot}+\bar{y})^2+\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij\cdot})^2\right]}{\left[k\,h\,l-(k+h-1)\right]}}$$

$$F_g value = \frac{\left[kl\,\sum_{i=1}^{h}(\bar{y}_{i \cdot \cdot}-\bar{y})^2\right]/(h-1)}{\frac{\left[l\,\sum_{j=1}^{k}\sum_{i=1}^{h}(\bar{y}_{ij\cdot}-\bar{y}_{i\cdot\cdot}-\bar{y}_{\cdot j\cdot}+\bar{y})^2+\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij\cdot})^2\right]}{\left[k\,h\,l-(k+h-1)\right]}}$$

`Pr(>F)`

$$P(F_{k-1,\,k\,h\,l-(k+h-1)} \geq F_f value)$$
$$P(F_{h-1,\,k\,h\,l-(k+h-1))} \geq F_g value)$$

- **Examples:**

```
> y <- c(1, 4, 10, 2.1, 3.5, 5.6, 8.4, 12, 6.5, 2, 1.2, 3.4)
> f <- factor(rep(letters[1:2], each = 6))
> f

 [1] a a a a a a b b b b b b
Levels: a b

> g <- factor(rep(LETTERS[2:1], times = 6))
> g

 [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 <- 12
> k <- 2
> h <- 2
> l <- 3
> l

[1] 3

> modello <- lm(formula = y ~ f + g)
> anova(object = modello)

Analysis of Variance Table

Response: y
          Df  Sum Sq Mean Sq F value Pr(>F)
f          1   4.441   4.441  0.2913 0.6025
g          1   0.188   0.188  0.0123 0.9141
Residuals  9 137.194  15.244

> res <- anova(object = modello)
> res$Df

[1] 1 1 9

> res$"Sum Sq"

[1]   4.440833   0.187500 137.194167

> res$"Mean Sq"

[1]  4.440833  0.187500 15.243796

> res$"F value"
```

```
[1] 0.29132070 0.01230009         NA

> res$"Pr(>F)"

[1] 0.6024717 0.9141250        NA
```

- **Note:** Il numero di replicazioni per cella l deve essere maggiore od uguale ad uno.

Anova a due fattori con interazione

- **Package:** `stats`
- **Sintassi:** `anova()`
- **Input:**

  `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`

f	$k-1$
g	$h-1$
f : *g*	$(k-1)\,(h-1)$
Residuals	$k\,h\,(l-1)$

  `Sum Sq`

f	$hl\,\sum_{j=1}^{k}(\bar{y}_{\cdot j\cdot}-\bar{y})^2$
g	$kl\,\sum_{i=1}^{h}(\bar{y}_{i\cdot\cdot}-\bar{y})^2$
f : *g*	$l\,\sum_{j=1}^{k}\sum_{i=1}^{h}(\bar{y}_{ij\cdot}-\bar{y}_{i\cdot\cdot}-\bar{y}_{\cdot j\cdot}+\bar{y})^2$
Residuals	$\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij\cdot})^2$

  `Mean Sq`

f	$\left[hl\,\sum_{j=1}^{k}(\bar{y}_{\cdot j\cdot}-\bar{y})^2\right]/\,(k-1)$
g	$\left[kl\,\sum_{i=1}^{h}(\bar{y}_{i\cdot\cdot}-\bar{y})^2\right]/\,(h-1)$
f : *g*	$\left[l\,\sum_{j=1}^{k}\sum_{i=1}^{h}(\bar{y}_{ij\cdot}-\bar{y}_{i\cdot\cdot}-\bar{y}_{\cdot j\cdot}+\bar{y})^2\right]/\,[(k-1)\,(h-1)]$
Residuals	$\left[\sum_{j=1}^{k}\sum_{i=1}^{h}\sum_{m=1}^{l}(y_{ijm}-\bar{y}_{ij\cdot})^2\right]/\,[k\,h\,(l-1)]$

F value

$$F_f value = \frac{\left[h\, l\, \sum_{j=1}^{k} (\bar{y}_{\cdot j \cdot} - \bar{y})^2\right] / (k-1)}{\left[\sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij\cdot})^2\right] / [k\, h\, (l-1)]}$$

$$F_g value = \frac{\left[kl\, \sum_{i=1}^{h} (\bar{y}_{i \cdot \cdot} - \bar{y})^2\right] / (h-1)}{\left[\sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij\cdot})^2\right] / [k\, h\, (l-1)]}$$

$$F_{f:g} value = \frac{\left[l\, \sum_{j=1}^{k} \sum_{i=1}^{h} (\bar{y}_{ij\cdot} - \bar{y}_{i\cdot\cdot} - \bar{y}_{\cdot j \cdot} + \bar{y})^2\right] / [(k-1)\,(h-1)]}{\left[\sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij\cdot})^2\right] / [kh\, (l-1)]}$$

Pr(>F)

$$P(F_{k-1,\, k\, h\, (l-1)} \geq F_f value)$$
$$P(F_{h-1,\, k\, h\, (l-1)} \geq F_g value)$$
$$P(F_{(k-1)\, (h-1),\, k\, h\, (l-1)}) \geq F_{f:g} value)$$

- **Examples:**

```
> y <- c(1, 4, 10, 2.1, 3.5, 5.6, 8.4, 12, 6.5, 2, 1.2, 3.4)
> f <- factor(rep(letters[1:2], each = 6))
> f

 [1] a a a a a a b b b b b b
Levels: a b

> g <- factor(rep(LETTERS[2:1], times = 6))
> g

 [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 <- 12
> k <- 2
> h <- 2
> l <- 3
> modello <- lm(formula = y ~ f + g + f:g)
> anova(object = modello)

Analysis of Variance Table

Response: y
          Df  Sum Sq Mean Sq F value Pr(>F)
f          1   4.441   4.441  0.2616 0.6228
g          1   0.188   0.188  0.0110 0.9189
f:g        1   1.401   1.401  0.0825 0.7812
Residuals  8 135.793  16.974

> res <- anova(object = modello)
> res$Df

[1] 1 1 1 8

> res$"Sum Sq"

[1]   4.440833   0.187500   1.400833 135.793333
```

```
> res$"Mean Sq"

[1]  4.440833  0.187500  1.400833 16.974167

> res$"F value"

[1] 0.26162305 0.01104620 0.08252737         NA

> res$"Pr(>F)"

[1] 0.6228225 0.9188831 0.7812018        NA
```

- **Note:** Il numero di replicazioni per cella l deve essere maggiore di uno.

7.3 Comandi utili in analisi della varianza

factor()

- **Package:** base
- **Input:**

  x vettore alfanumerico di dimensione n

  levels etichette di livello

  labels etichette di livello

  ordered = TRUE / FALSE livelli su scala ordinale

- **Description:** crea un fattore
- **Examples:**

```
> factor(x = rep(c("U", "D"), each = 4), levels = c("U", "D"))

[1] U U U U D D D D
Levels: U D

> factor(x = rep(c("U", "D"), each = 4), levels = c("D", "U"))

[1] U U U U D D D D
Levels: D U

> factor(x = rep(1:2, each = 4), labels = c("U", "D"))

[1] U U U U D D D D
Levels: U D

> factor(x = rep(1:2, each = 4), labels = c("D", "U"))

[1] D D D D U U U U
Levels: D U

> factor(x = rep(1:2, each = 4), labels = c("U", "D"), ordered = TRUE)

[1] U U U U D D D D
Levels: U < D

> factor(x = rep(1:2, each = 4), labels = c("D", "U"), ordered = TRUE)

[1] D D D D U U U U
Levels: D < U
```

```
> factor(x = rep(c("U", "D"), each = 4), levels = c("U", "D"),
+     ordered = TRUE)

[1] U U U U D D D D
Levels: U < D

> factor(x = rep(c("U", "D"), each = 4), levels = c("D", "U"),
+     ordered = TRUE)

[1] U U U U D D D D
Levels: D < U

> fattore <- factor(x = scan(what = "character"))
```

as.factor()

- **Package:** base
- **Input:**

  x vettore alfanumerico di dimensione n

- **Description:** creazione di un fattore
- **Examples:**

```
> x <- c("a", "b", "b", "c", "a", "c", "b", "b", "c", "a", "c",
+     "a")
> as.factor(x)

 [1] a b b c a c b b c a c a
Levels: a b c

> x <- c("ALTO", "ALTO", "BASSO", "MEDIO", "ALTO", "BASSO", "MEDIO",
+     "BASSO")
> as.factor(x)

[1] ALTO  ALTO  BASSO MEDIO ALTO  BASSO MEDIO BASSO
Levels: ALTO BASSO MEDIO
```

relevel()

- **Package:** stats
- **Input:**

  x fattore a k livelli

  ref livello di riferimento

- **Description:** ricodificazione dei livelli di un fattore
- **Examples:**

```
> x <- factor(c("a", "b", "c", "a", "b", "b", "c", "c", "a", "b"))
> x

 [1] a b c a b b c c a b
Levels: a b c

> relevel(x, ref = "b")
```

```
 [1] a b c a b b c c a b
Levels: b a c
```

```
> relevel(x, ref = "c")
```

```
 [1] a b c a b b c c a b
Levels: c a b
```

levels()

- **Package:** base
- **Input:**

  f fattore a k livelli

- **Description:** nome dei livelli
- **Examples:**

```
> f <- factor(rep(1:2, each = 5))
> f
```

```
 [1] 1 1 1 1 1 2 2 2 2 2
Levels: 1 2
```

```
> levels(f)
```

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

```
> f <- factor(rep(c("U", "D"), each = 4))
> f
```

```
[1] U U U U D D D D
Levels: D U
```

```
> levels(f)
```

```
[1] "D" "U"
```

nlevels()

- **Package:** base
- **Input:**

  f fattore a k livelli

- **Description:** numero di livelli
- **Examples:**

```
> f <- factor(rep(1:2, each = 5))
> f
```

```
 [1] 1 1 1 1 1 2 2 2 2 2
Levels: 1 2
```

```
> nlevels(f)
```

```
[1] 2
```

```
> f <- factor(c("A", "A", "A", "A", "B", "B", "B", "B", "C", "C"))
> f
```

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

```
> nlevels(f)
```

```
[1] 3
```

ordered()

- **Package:** base
- **Input:**

  x vettore alfanumerico di dimensione n

  levels etichette dei livelli

- **Description:** fattore con livelli su scala ordinale
- **Examples:**

```
> ordered(x = c(rep("U", 5), rep("D", 5)), levels = c("U", "D"))
```

```
 [1] U U U U U D D D D D
Levels: U < D
```

```
> ordered(x = c(rep("U", 5), rep("D", 5)), levels = c("D", "U"))
```

```
 [1] U U U U U D D D D D
Levels: D < U
```

```
> fattore <- ordered(x = c("a", "b", "c", "a", "b", "b", "c", "c",
+     "a", "b"), levels = c("a", "b", "c"))
> fattore
```

```
 [1] a b c a b b c c a b
Levels: a < b < c
```

```
> fattore < "b"
```

```
 [1]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE
```

as.ordered()

- **Package:** base
- **Input:**

  x vettore alfanumerico di dimensione n

- **Description:** fattore con livelli su scala ordinale
- **Examples:**

```
> as.ordered(x = c(rep("U", 5), rep("D", 5)))

 [1] U U U U U D D D D D
Levels: D < U

> as.ordered(x = c(rep("U", 5), rep("D", 5)))

 [1] U U U U U D D D D D
Levels: D < U

> as.ordered(x = c("a", "b", "c", "a", "b", "b", "c", "c", "a",
+     "b"))

 [1] a b c a b b c c a b
Levels: a < b < c
```

letters[ ]

- **Package:** base
- **Description:** lettere minuscole
- **Examples:**

```
> letters[1:6]

[1] "a" "b" "c" "d" "e" "f"

> letters[c(3, 5, 6, 26)]

[1] "c" "e" "f" "z"
```

LETTERS[ ]

- **Package:** base
- **Description:** lettere maiuscole
- **Examples:**

```
> LETTERS[1:6]

[1] "A" "B" "C" "D" "E" "F"

> LETTERS[c(3, 5, 6, 26)]

[1] "C" "E" "F" "Z"
```

as.numeric()

- **Package:** base
- **Input:**

  x fattore a k livelli

- **Description:** codici dei livelli
- **Examples:**

```
> x <- factor(c(2, 3, 1, 1, 1, 3, 4, 4, 1, 2), labels = 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

> x <- factor(c("M", "F", "M", "F", "M", "F", "F", "M"), levels = c("M",
+     "F"))
> x

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

> as.numeric(x)

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

as.integer()

- **Package:** base
- **Input:**

  x fattore a k livelli

- **Description:** codici dei livelli
- **Examples:**

```
> x <- factor(c(2, 3, 1, 1, 1, 3, 4, 4, 1, 2), labels = 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

> x <- factor(c("M", "F", "M", "F", "M", "F", "F", "M"), levels = c("M",
+     "F"))
> x

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

> as.integer(x)

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

unclass()

- **Package:** base
- **Input:**

  `x` fattore a k livelli

- **Description:** codici dei livelli
- **Examples:**

```
> x <- factor(c(2, 3, 1, 1, 1, 3, 4, 4, 1, 2), labels = c("A",
+     "B", "C", "D"))
> x

 [1] B C A A A C D D A B
Levels: A B C D

> unclass(x)

 [1] 2 3 1 1 1 3 4 4 1 2
attr(,"levels")
[1] "A" "B" "C" "D"

> x <- factor(c("M", "F", "M", "F", "M", "F", "F", "M"), levels = c("M",
+     "F"))
> x

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

> unclass(x)

[1] 1 2 1 2 1 2 2 1
attr(,"levels")
[1] "M" "F"
```

by()

- **Package:** base
- **Input:**

  `data` vettore numerico y di dimensione n

  `INDICES` fattore f a k livelli

  `FUN` funzione

- **Description:** applica `FUN` ad ogni vettore numerico per livello del fattore
- **Example 1:**

```
> y <- c(1.2, 2.3, 5.6, 3.5, 2.5, 3.8, 6.8, 5.7, 3.7, 6.4)
> f <- factor(c("a", "b", "c", "a", "b", "b", "c", "c", "a", "b"))
> f

 [1] a b c a b b c c a b
Levels: a b c

> by(data = y, INDICES = f, FUN = mean)
```

```
f: a
[1] 2.8
------------------------------------------------------------
f: b
[1] 3.75
------------------------------------------------------------
f: c
[1] 6.033333
```

- **Example 2:**

```
> y <- c(1.2, 2.3, 5.6, 3.5, 2.5, 3.8, 6.8, 5.7, 3.7, 6.4)
> g <- factor(c("alto", "medio", "basso", "alto", "medio", "basso",
+     "medio", "alto", "alto", "basso"))
> g

 [1] alto  medio basso alto  medio basso medio alto  alto  basso
Levels: alto basso medio

> by(data = y, INDICES = g, FUN = mean)

g: alto
[1] 3.525
------------------------------------------------------------
g: basso
[1] 5.266667
------------------------------------------------------------
g: medio
[1] 3.866667
```

- **Example 3:**

```
> y <- c(1.2, 2.3, 5.6, 3.5, 2.5, 3.8, 6.8, 5.7, 3.7, 6.4)
> f <- factor(c("a", "b", "c", "a", "b", "b", "c", "c", "a", "b"))
> f

 [1] a b c a b b c c a b
Levels: a b c

> g <- factor(c("alto", "medio", "basso", "alto", "medio", "basso",
+     "medio", "alto", "alto", "basso"))
> g

 [1] alto  medio basso alto  medio basso medio alto  alto  basso
Levels: alto basso medio

> 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
- **Input:**

  X vettore numerico x di dimensione n

  INDEX fattore f a k livelli

  FUN funzione
- **Description:** applica la funzione FUN ad ogni gruppo di elementi di x definito dai livelli di f
- **Examples:**

```
> X <- c(1.2, 2.3, 5.6, 3.5, 2.5, 3.8, 6.8, 5.7, 3.7, 6.4)
> f <- factor(c("a", "b", "c", "a", "b", "b", "c", "c", "a", "b"))
> f

 [1] a b c a b b c c a b
Levels: a b c

> g <- factor(c("alto", "medio", "basso", "alto", "medio", "basso",
+      "medio", "alto", "alto", "basso"))
> g

 [1] alto  medio basso alto  medio basso medio alto  alto  basso
Levels: alto basso medio

> tapply(X, INDEX = f, FUN = mean)

       a        b        c
2.800000 3.750000 6.033333

> tapply(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
- **Input:**

  `n` numero dei livelli

  `k` numero delle replicazioni

  `length` dimensione del fattore risultato

  `labels` nomi dei livelli

  `ordered = TRUE / FALSE` fattore ordinato
- **Description:** crea un fattore
- **Examples:**

```
> 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

> gl(n = 2, k = 8, labels = c("Control", "Treat"), ordered = TRUE)

 [1] Control Control Control Control Control Control Control Control Treat
[10] Treat   Treat   Treat   Treat   Treat   Treat   Treat
Levels: Control < Treat
```

ave()

- **Package:** stats
- **Input:**

  `x` vettore numerico di dimensione n

  `f` fattore a k livelli di dimensione n

  `FUN` funzione
- **Description:** applica e replica la funzione FUN ad ogni gruppo di elementi di x definito dai livelli di f
- **Examples:**

```
> x <- c(1, 2, 3, 4, 5, 6, 7, 8)
> f <- factor(rep(letters[1:2], each = 4))
> 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 <- c(1, 2, 3, 4, 5, 6, 7, 8)
> f <- factor(rep(letters[1:2], each = 4))
> 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
```

```
> x <- c(1, 2, 3, 4, 5, 6, 7, 8)
> f <- factor(rep(letters[1:2], each = 4))
> 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 = function(x) mean(x, trim = 0.1))
```

```
[1] 2.5 2.5 2.5 2.5 6.5 6.5 6.5 6.5
```

cut()

- **Package:** base
- **Input:**

  `x` vettore numerico di dimensione n

  `breaks` estremi delle classi di ampiezza b_i

  `right = TRUE / FALSE` classi chiuse a destra $\left(a_{(i)},\, a_{(i+1)}\right]$ oppure a sinistra $\left[a_{(i)},\, a_{(i+1)}\right)$

  `include.lowest = TRUE / FALSE` estremo incluso

  `labels` etichette

  `ordered_result = TRUE / FALSE` fattore ordinato

- **Description:** raggruppamento in classi
- **Examples:**

```
> x <- c(1.2, 2.3, 4.5, 5.4, 3.4, 5.4, 2.3, 2.1, 1.23, 4.3, 0.3)
> n <- 11
> cut(x, breaks = c(0, 4, 6), right = TRUE, include.lowest = FALSE,
+     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 <- c(1, 2, 3, 4, 5.6, 7.4, 1.2, 4, 4.4)
> n <- 9
> cut(x, breaks = c(0, 4, 8), right = TRUE, include.lowest = FALSE,
+     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

> x <- c(1, 2, 3, 4, 5.6, 7.4, 1.2, 4, 4.4)
> n <- 9
> cut(x, breaks = c(0, 4, 8), right = TRUE, include.lowest = FALSE,
+     labels = c("0-4", "4-8"), ordered_result = TRUE)

[1] 0-4 0-4 0-4 0-4 4-8 4-8 0-4 0-4 4-8
Levels: 0-4 < 4-8
```

summary()

- **Package:** base
- **Input:**

  `object` fattore a k livelli di dimensione n

- **Description:** distribuzione di frequenza assoluta
- **Examples:**

```
> f <- factor(c("a", "b", "b", "c", "a", "c", "b", "b", "c", "a",
+     "c", "a"))
> f

 [1] a b b c a c b b c a c a
Levels: a b c

> summary(object = f)
```

```
a b c
4 4 4

> f <- factor(c("ALTO", "ALTO", "BASSO", "MEDIO", "ALTO", "BASSO",
+     "MEDIO", "BASSO"))
> f

[1] ALTO  ALTO  BASSO MEDIO ALTO  BASSO MEDIO BASSO
Levels: ALTO BASSO MEDIO

> summary(object = f)

 ALTO BASSO MEDIO
    3     3     2
```

interaction()

- **Package:** base
- **Input:**

  ... fattori su cui eseguire l'interazione
- **Description:** interazione tra fattori
- **Example 1:**

```
> a <- factor(rep(1:2, each = 4))
> a

[1] 1 1 1 1 2 2 2 2
Levels: 1 2

> b <- factor(rep(c("ctrl", "treat"), times = 2, each = 2))
> b

[1] ctrl  ctrl  treat treat ctrl  ctrl  treat treat
Levels: ctrl treat

> interaction(a, b)

[1] 1.ctrl  1.ctrl  1.treat 1.treat 2.ctrl  2.ctrl  2.treat 2.treat
Levels: 1.ctrl 2.ctrl 1.treat 2.treat
```

- **Example 2:**

```
> a <- factor(rep(1:2, each = 4))
> a

[1] 1 1 1 1 2 2 2 2
Levels: 1 2

> b <- factor(rep(c("M", "F"), times = 4))
> b

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

> interaction(a, b)
```

```
[1] 1.M 1.F 1.M 1.F 2.M 2.F 2.M 2.F
Levels: 1.F 2.F 1.M 2.M
```

- **Example 3:**

```
> a <- factor(rep(c("M", "F"), times = 4))
> a

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

> b <- factor(rep(c("M", "F"), times = 4))
> b

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

> interaction(a, b)

[1] M.M F.F M.M F.F M.M F.F M.M F.F
Levels: F.F M.F F.M M.M
```

expand.grid()

- **Package:** base
- **Input:**

  . . . vettori numerici o fattori

- **Description:** creazione di un data frame da tutte le combinazioni di vettori numerici o fattori
- **Example 1:**

```
> height <- c(60, 80)
> weight <- c(100, 300, 500)
> sex <- factor(c("Male", "Female"))
> mydf <- expand.grid(height = height, weight = weight, sex = sex)
> mydf

   height weight    sex
1      60    100   Male
2      80    100   Male
3      60    300   Male
4      80    300   Male
5      60    500   Male
6      80    500   Male
7      60    100 Female
8      80    100 Female
9      60    300 Female
10     80    300 Female
11     60    500 Female
12     80    500 Female

> is.data.frame(mydf)

[1] TRUE
```

- **Example 2:**

```
> Sex <- factor(c("Women", "Men"), levels = c("Women", "Men"))
> Age <- factor(c("18-23", "24-40", ">40"), levels = c("18-23",
+     "24-40", ">40"))
> Response <- factor(c("little importance", "importance", "very importance"),
+     levels = c("little importance", "importance", "very importance"))
> mydf <- expand.grid(Sex = Sex, Age = Age, Response = Response)
> Freq <- c(26, 40, 9, 17, 5, 8, 12, 17, 21, 15, 14, 15, 7, 8,
+     15, 12, 41, 18)
> mydf <- cbind(mydf, Freq)
> mydf
```

```
     Sex   Age          Response Freq
1  Women 18-23 little importance   26
2    Men 18-23 little importance   40
3  Women 24-40 little importance    9
4    Men 24-40 little importance   17
5  Women   >40 little importance    5
6    Men   >40 little importance    8
7  Women 18-23        importance   12
8    Men 18-23        importance   17
9  Women 24-40        importance   21
10   Men 24-40        importance   15
11 Women   >40        importance   14
12   Men   >40        importance   15
13 Women 18-23   very importance    7
14   Men 18-23   very importance    8
15 Women 24-40   very importance   15
16   Men 24-40   very importance   12
17 Women   >40   very importance   41
18   Men   >40   very importance   18
```

```
> is.data.frame(mydf)
```

```
[1] TRUE
```

- **Example 3:**

```
> x <- LETTERS[1:3]
> y <- 1:2
> z <- letters[1:2]
> mydf <- expand.grid(x = x, y = y, z = z)
> mydf
```

```
   x y z
1  A 1 a
2  B 1 a
3  C 1 a
4  A 2 a
5  B 2 a
6  C 2 a
7  A 1 b
8  B 1 b
9  C 1 b
10 A 2 b
11 B 2 b
12 C 2 b
```

```
> is.data.frame(mydf)
```

```
[1] TRUE
```

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

- **Package:** `stats`
- **Sintassi:** `TukeyHSD()`
- **Input:**

  `y` vettore numerico di dimensione n

  `f` fattore con livelli $1, 2, \ldots, k$

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `f` intervallo di confidenza a livello $1 - \alpha$ per il fattore `f`

- **Formula:**

  `f`

$$\bar{y}_i - \bar{y}_j \quad \forall\, i > j \,=\, 1, 2, \ldots, k$$

$$\bar{y}_i - \bar{y}_j \,\mp\, q_{1-\alpha,\, k,\, n-k}\, s_P \sqrt{1 \,/\, (2\, n_i) + 1 \,/\, (2\, n_j)} \quad \forall\, i > j \,=\, 1, 2, \ldots, k$$

$$\text{dove} \quad s_P^2 = \sum_{j=1}^{k} \sum_{i=1}^{n_j} (y_{ij} - \bar{y}_j)^2 \,/\, (n - k)$$

- **Examples:**

```
> y <- c(19, 24, 24, 27, 20, 24, 22, 21, 22, 29, 18, 17)
> f <- factor(rep(1:3, times = 4))
> f

 [1] 1 2 3 1 2 3 1 2 3 1 2 3
Levels: 1 2 3

> n <- 12
> k <- 3
> alpha <- 0.05
> qTUKEY <- qtukey(0.95, nmeans = k, df = n - k)
> qTUKEY

[1] 3.948492

> TukeyHSD(aov(formula = y ~ f), conf.level = 0.95)

  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = y ~ f)

$f
    diff        lwr      upr     p adj
2-1 -3.5 -10.534094 3.534094 0.3860664
3-1 -2.5  -9.534094 4.534094 0.5996130
3-2  1.0  -6.034094 8.034094 0.9175944

> res <- TukeyHSD(aov(formula = y ~ f), conf.level = 0.95)
> y1m <- mean(y[f == "1"])
> y1m

[1] 24.25

> y2m <- mean(y[f == "2"])
> y2m

[1] 20.75

> y3m <- mean(y[f == "3"])
> y3m

[1] 21.75

> differ <- c(y2m - y1m, y3m - y1m, y3m - y2m)
> n1 <- length(y[f == "1"])
> n1

[1] 4

> n2 <- length(y[f == "2"])
> n2
```

```
[1] 4

> n3 <- length(y[f == "3"])
> n3

[1] 4

> Sp2 <- anova(lm(formula = y ~ f))$"Mean Sq"[2]
> stderror <- sqrt(Sp2) * sqrt(c(1/(2 * n2) + 1/(2 * n1), 1/(2 *
+     n3) + 1/(2 * n1), 1/(2 * n3) + 1/(2 * n2)))
> lower <- differ - qTUKEY * stderror
> upper <- differ + qTUKEY * stderror
> matrix(data = cbind(differ, lower, upper), nrow = 3, ncol = 3,
+     dimnames = list(c("2-1", "3-1", "3-2"), c("diff", "lwr",
+         "upr")))

    diff        lwr      upr
2-1 -3.5 -10.534094 3.534094
3-1 -2.5  -9.534094 4.534094
3-2  1.0  -6.034094 8.034094

> res$f

    diff        lwr      upr     p adj
2-1 -3.5 -10.534094 3.534094 0.3860664
3-1 -2.5  -9.534094 4.534094 0.5996130
3-2  1.0  -6.034094 8.034094 0.9175944
```

- **Note:** Il numero di confronti è pari a $\binom{k}{2}$ per il fattore `f`.

Applicazione in Anova a due fattori senza interazione

- **Package:** `stats`
- **Sintassi:** `TukeyHSD()`
- **Input:**

  `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`

$$\bar{y}_{\cdot i \cdot} - \bar{y}_{\cdot j \cdot} \quad \forall\, i > j \,=\, 1, 2, \ldots, k$$

$$\bar{y}_{\cdot i \cdot} - \bar{y}_{\cdot j \cdot} \mp q_{1-\alpha,\, k,\, k\,h\,l-(k+h-1)}\, s_P \,/\, \sqrt{h\,l} \quad \forall\, i > j \,=\, 1, 2, \ldots, k$$

$$\text{dove} \quad s_P^2 \,=\, \frac{l \sum_{j=1}^{k} \sum_{i=1}^{h} (\bar{y}_{ij\cdot} - \bar{y}_{i\cdot\cdot} - \bar{y}_{\cdot j \cdot} + \bar{y})^2 + \sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij\cdot})^2}{k\,h\,l - (k + h - 1)}$$

g

$$\bar{y}_{i\cdot\cdot} - \bar{y}_{j\cdot\cdot} \quad \forall\, i > j \,=\, 1, 2, \dots, h$$

$$\bar{y}_{i\cdot\cdot} - \bar{y}_{j\cdot\cdot} \mp q_{1-\alpha,\, h,\, k\, h\, l-(k+h-1)}\, s_P \,/\, \sqrt{k\, l} \quad \forall\, i > j \,=\, 1, 2, \dots, h$$

$$\text{dove} \quad s_P^2 \,=\, \frac{l\,\sum_{j=1}^{k}\,\sum_{i=1}^{h}\,(\bar{y}_{ij\cdot} - \bar{y}_{i\cdot\cdot} - \bar{y}_{\cdot j\cdot} + \bar{y})^2 + \sum_{j=1}^{k}\,\sum_{i=1}^{h}\,\sum_{m=1}^{l}\,(y_{ijm} - \bar{y}_{ij\cdot})^2}{k\, h\, l - (k + h - 1)}$$

- **Examples:**

```
> y <- c(1, 4, 10, 2.1, 3.5, 5.6, 8.4, 12, 16.5, 22, 1.2, 3.4)
> f <- factor(rep(letters[1:2], each = 6))
> f

 [1] a a a a a a b b b b b b
Levels: a b

> g <- factor(rep(LETTERS[2:1], times = 6))
> g

 [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 <- 12
> k <- 2
> h <- 2
> l <- 3
> alpha <- 0.05
> qTUKEYf <- qtukey(0.95, nmeans = k, df = k * h * l - (k + h -
+     1))
> qTUKEYf

[1] 3.199173

> qTUKEYg <- qtukey(0.95, nmeans = h, df = k * h * l - (k + h -
+     1))
> qTUKEYg

[1] 3.199173

> TukeyHSD(aov(formula = y ~ f + g), conf.level = 0.95)

  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = y ~ f + g)

$f
        diff       lwr      upr     p adj
b-a 6.216667 -2.001707 14.43504 0.1212097

$g
         diff      lwr      upr     p adj
B-A -1.416667 -9.63504 6.801707 0.7056442
```

```
> res <- TukeyHSD(aov(formula = y ~ f + g), conf.level = 0.95)
> y.1.m <- mean(y[f == "a"])
> y.1.m

[1] 4.366667

> y.2.m <- mean(y[f == "b"])
> y.2.m

[1] 10.58333

> differ <- y.2.m - y.1.m
> Sp2 <- anova(lm(formula = y ~ f + g))$"Mean Sq"[3]
> stderror <- sqrt(Sp2)/sqrt(h * l)
> lower <- differ - qTUKEYf * stderror
> upper <- differ + qTUKEYf * stderror
> matrix(data = cbind(differ, lower, upper), nrow = 1, ncol = 3,
+     dimnames = list("b-a", c("diff", "lwr", "upr")))

        diff       lwr      upr
b-a 6.216667 -2.001707 14.43504

> res$f

        diff       lwr      upr     p adj
b-a 6.216667 -2.001707 14.43504 0.1212097

> y1..m <- mean(y[g == "A"])
> y1..m

[1] 8.183333

> y2..m <- mean(y[g == "B"])
> y2..m

[1] 6.766667

> differ <- y2..m - y1..m
> Sp2 <- anova(lm(formula = y ~ f + g))$"Mean Sq"[3]
> stderror <- sqrt(Sp2)/sqrt(k * l)
> lower <- differ - qTUKEYg * stderror
> upper <- differ + qTUKEYg * stderror
> matrix(data = cbind(differ, lower, upper), nrow = 1, ncol = 3,
+     dimnames = list("B-A", c("diff", "lwr", "upr")))

         diff      lwr      upr
B-A -1.416667 -9.63504 6.801707

> res$g

         diff      lwr      upr     p adj
B-A -1.416667 -9.63504 6.801707 0.7056442
```

- **Note 1:** Il numero di replicazioni per cella l deve essere maggiore od uguale ad uno.
- **Note 2:** Il numero di confronti è pari a $\binom{k}{2}$ per il fattore `f`.
- **Note 3:** Il numero di confronti è pari a $\binom{h}{2}$ per il fattore `g`.

Applicazione in Anova a due fattori con interazione

- **Package:** stats
- **Sintassi:** TukeyHSD()
- **Input:**

  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

  f:g intervallo di confidenza a livello $1 - \alpha$ per l'interazione f:g

- **Formula:**

  f

$$\bar{y}_{\cdot i \cdot} - \bar{y}_{\cdot j \cdot} \quad \forall\, i > j \,=\, 1, 2, \ldots, k$$

$$\bar{y}_{\cdot i \cdot} - \bar{y}_{\cdot j \cdot} \mp q_{1-\alpha,\, k,\, k\, h\, (l-1)}\, s_P \,/\, \sqrt{h\, l} \quad \forall\, i > j \,=\, 1, 2, \ldots, k$$

$$\text{dove} \quad s_P^2 \,=\, \sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij\cdot})^2 \,/\, [k\, h\, (l-1)]$$

  g

$$\bar{y}_{i \cdot \cdot} - \bar{y}_{j \cdot \cdot} \quad \forall\, i > j \,=\, 1, 2, \ldots, h$$

$$\bar{y}_{i \cdot \cdot} - \bar{y}_{j \cdot \cdot} \mp q_{1-\alpha,\, h,\, k\, h\, (l-1)}\, s_P \,/\, \sqrt{k\, l} \quad \forall\, i > j \,=\, 1, 2, \ldots, h$$

$$\text{dove} \quad s_P^2 \,=\, \sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij\cdot})^2 \,/\, [k\, h\, (l-1)]$$

  f:g

$$\bar{y}_{ij\cdot} - \bar{y}_{uw\cdot} \quad \forall\, i,\, u \,=\, 1, 2, \ldots, h \quad \forall\, j,\, w \,=\, 1, 2, \ldots, k$$

$$\bar{y}_{ij\cdot} - \bar{y}_{uw\cdot} \mp q_{1-\alpha,\, k\,h,\, k\, h\, (l-1)}\, s_P \,/\, \sqrt{l} \quad \forall\, i,\, u \,=\, 1, 2, \ldots, h \quad \forall\, j,\, w \,=\, 1, 2, \ldots, k$$

$$\text{dove} \quad s_P^2 \,=\, \sum_{j=1}^{k} \sum_{i=1}^{h} \sum_{m=1}^{l} (y_{ijm} - \bar{y}_{ij\cdot})^2 \,/\, [k\, h\, (l-1)]$$

- **Examples:**

```
> y <- c(1, 4, 10, 2.1, 3.5, 5.6, 8.4, 12, 16.5, 22, 1.2, 3.4)
> f <- factor(rep(letters[1:2], each = 6))
> f

 [1] a a a a a a b b b b b b
Levels: a b

> g <- factor(rep(LETTERS[1:2], times = 6))
> g

 [1] A B A B A B A B A B A B
Levels: A B

> table(f, g)
```

```
   g
f   A B
  a 3 3
  b 3 3

> n <- 12
> k <- 2
> h <- 2
> l <- 3
> alpha <- 0.05
> qTUKEYf <- qtukey(0.95, nmeans = k, df = k * h * (l - 1))
> qTUKEYf

[1] 3.261182

> qTUKEYg <- qtukey(0.95, nmeans = h, df = k * h * (l - 1))
> qTUKEYg

[1] 3.261182

> qTUKEYfg <- qtukey(0.95, nmeans = k * h, df = k * h * (l - 1))
> qTUKEYfg

[1] 4.52881

> TukeyHSD(aov(y ~ f + g + f:g), conf.level = 0.95)

  Tukey multiple comparisons of means
    95% family-wise confidence level

Fit: aov(formula = y ~ f + g + f:g)

$f
        diff       lwr      upr     p adj
b-a 6.216667 -2.460179 14.89351 0.1371018

$g
        diff      lwr      upr     p adj
B-A 1.416667 -7.26018 10.09351 0.7163341

$`f:g`
              diff        lwr      upr     p adj
b:A-a:A  3.8666667 -13.173972 20.90731 0.8838028
a:B-a:A -0.9333333 -17.973972 16.10731 0.9979198
b:B-a:A  7.6333333  -9.407306 24.67397 0.5144007
a:B-b:A -4.8000000 -21.840639 12.24064 0.8043752
b:B-b:A  3.7666667 -13.273972 20.80731 0.8912420
b:B-a:B  8.5666667  -8.473972 25.60731 0.4251472

> res <- TukeyHSD(aov(y ~ f + g + f:g), conf.level = 0.95)
> y.1.m <- mean(y[f == "a"])
> y.1.m

[1] 4.366667

> y.2.m <- mean(y[f == "b"])
> y.2.m

[1] 10.58333
```

```
> differ <- y.2.m - y.1.m
> Sp2 <- anova(lm(formula = y ~ f + g))$"Mean Sq"[4]
> stderror <- sqrt(Sp2)/sqrt(h * l)
> lower <- differ - qTUKEYf * stderror
> upper <- differ + qTUKEYf * stderror
> matrix(data = cbind(differ, lower, upper), nrow = 1, ncol = 3,
+     dimnames = list("b-a", c("diff", "lwr", "upr")))

        diff lwr upr
b-a 6.216667  NA  NA

> res$f

        diff       lwr      upr     p adj
b-a 6.216667 -2.460179 14.89351 0.1371018

> y1..m <- mean(y[g == "A"])
> y1..m

[1] 6.766667

> y2..m <- mean(y[g == "B"])
> y2..m

[1] 8.183333

> differ <- y2..m - y1..m
> Sp2 <- anova(lm(formula = y ~ f + g))$"Mean Sq"[3]
> stderror <- sqrt(Sp2)/sqrt(k * l)
> lower <- differ - qTUKEYg * stderror
> upper <- differ + qTUKEYg * stderror
> matrix(data = cbind(differ, lower, upper), nrow = 1, ncol = 3,
+     dimnames = list("B-A", c("diff", "lwr", "upr")))

        diff       lwr      upr
B-A 1.416667 -6.961002 9.794335

> res$g

        diff      lwr      upr     p adj
B-A 1.416667 -7.26018 10.09351 0.7163341

> y11.m <- mean(y[f == "a" & g == "A"])
> y11.m

[1] 4.833333

> y12.m <- mean(y[f == "b" & g == "A"])
> y12.m

[1] 8.7

> y21.m <- mean(y[f == "a" & g == "B"])
> y21.m

[1] 3.9

> y22.m <- mean(y[f == "b" & g == "B"])
> y22.m
```

```
[1] 12.46667
```

```
> differ <- c(y12.m - y11.m, y21.m - y11.m, y22.m - y11.m, y21.m -
+     y12.m, y22.m - y12.m, y22.m - y21.m)
> Sp2 <- anova(lm(formula = y ~ f * g))$"Mean Sq"[4]
> stderror <- rep(sqrt(Sp2)/sqrt(l), times = 6)
> lower <- differ - qTUKEYfg * stderror
> upper <- differ + qTUKEYfg * stderror
> matrix(data = cbind(differ, lower, upper), nrow = 6, ncol = 3,
+     dimnames = list(c("b:A-a:A", "a:B-a:A", "b:B-a:A", "a:B-b:A",
+         "b:B-b:A", "b:B-a:B"), c("diff", "lwr", "upr")))
```

```
              diff        lwr      upr
b:A-a:A  3.8666667 -13.173972 20.90731
a:B-a:A -0.9333333 -17.973972 16.10731
b:B-a:A  7.6333333  -9.407306 24.67397
a:B-b:A -4.8000000 -21.840639 12.24064
b:B-b:A  3.7666667 -13.273972 20.80731
b:B-a:B  8.5666667  -8.473972 25.60731
```

```
> res$"f:g"
```

```
              diff        lwr      upr     p adj
b:A-a:A  3.8666667 -13.173972 20.90731 0.8838028
a:B-a:A -0.9333333 -17.973972 16.10731 0.9979198
b:B-a:A  7.6333333  -9.407306 24.67397 0.5144007
a:B-b:A -4.8000000 -21.840639 12.24064 0.8043752
b:B-b:A  3.7666667 -13.273972 20.80731 0.8912420
b:B-a:B  8.5666667  -8.473972 25.60731 0.4251472
```

- **Note 1:** Il numero di replicazioni per cella l deve essere maggiore di uno.
- **Note 2:** Il numero di confronti è pari a $\binom{k}{2}$ per il fattore f.
- **Note 3:** Il numero di confronti è pari a $\binom{h}{2}$ per il fattore g.
- **Note 4:** Il numero di confronti è pari a $\binom{k\,h}{2}$ per l'interazione f:g.

8.3 Metodo di Bonferroni

Applicazione in Anova ad un fattore

- **Package:** stats
- **Sintassi:** pairwise.t.test()
- **Input:**

  y vettore numerico di dimensione n

  f fattore con livelli $1, 2, \ldots, k$ livelli di dimensione n

  p.adjust.method = "bonferroni"
- **Output:**

  p.value p-value
- **Formula:**

p.value

$$2 \binom{k}{2} P(t_{n-k} \leq -|\,t\,|) \,=\, k\,(k-1)\,P(t_{n-k} \leq -|\,t\,|)$$

$$\text{dove} \quad t \,=\, \frac{\bar{y}_i - \bar{y}_j}{s_P \sqrt{1\,/\,n_i + 1\,/\,n_j}} \quad \forall\, i > j \,=\, 1,\, 2,\, \dots,\, k$$

$$\text{con } s_P^2 \,=\, \sum_{j=1}^{k} \sum_{i=1}^{n_j} (y_{ij} - \bar{y}_j)^2 \,/\, (n-k)$$

- **Examples:**

```
> y <- c(1, 14, 1, 12.1, 3.5, 5.6, 18.4, 12, 1.65, 22, 1.2, 1.34)
> f <- factor(rep(1:3, times = 4))
> f

 [1] 1 2 3 1 2 3 1 2 3 1 2 3
Levels: 1 2 3

> n <- 12
> k <- 3
> m.1 <- mean(y[f == "1"])
> m.2 <- mean(y[f == "2"])
> m.3 <- mean(y[f == "3"])
> n1 <- length(y[f == "1"])
> n2 <- length(y[f == "2"])
> n3 <- length(y[f == "3"])
> s2 <- anova(lm(formula = y ~ f))$"Mean Sq"[2]
> s <- sqrt(s2)
> t12 <- (m.2 - m.1)/(s * sqrt(1/n1 + 1/n2))
> t13 <- (m.3 - m.1)/(s * sqrt(1/n3 + 1/n1))
> t23 <- (m.3 - m.2)/(s * sqrt(1/n3 + 1/n2))
> p12 <- k * (k - 1) * pt(-abs(t12), df = n - k)
> p13 <- k * (k - 1) * pt(-abs(t13), df = n - k)
> p23 <- k * (k - 1) * pt(-abs(t23), df = n - k)
> matrix(data = c(p12, p13, NA, p23), dimnames = list(c("2", "3"),
+     c("1", "2")), nrow = 2, ncol = 2)

          1         2
2 0.7493036        NA
3 0.1258454 0.8521961

> pairwise.t.test(y, f, p.adjust.method = "bonferroni")

        Pairwise comparisons using t tests with pooled SD

data:  y and f

  1    2
2 0.75 -
3 0.13 0.85

P value adjustment method: bonferroni

> res <- pairwise.t.test(y, f, p.adjust.method = "bonferroni")
> res$p.value

          1         2
2 0.7493036        NA
3 0.1258454 0.8521961
```

8.4 Metodo di Student

Applicazione in Anova ad un fattore

- **Package:** stats
- **Sintassi:** `pairwise.t.test()`
- **Input:**

  `y` vettore numerico di dimensione n

  `f` fattore con livelli $1, 2, \ldots, k$ di dimensione n

  `p.adjust.method = "none"`

- **Output:**

  `p.value` p-value

- **Formula:**

  `p.value`

$$2\,P(t_{n-k} \leq -|\,t\,|)$$

$$\text{dove} \quad t \,=\, \frac{\bar{y}_i - \bar{y}_j}{s_P\,\sqrt{1\,/\,n_i + 1\,/\,n_j}} \quad \forall\, i > j \,=\, 1, 2, \ldots, k$$

$$\text{con } s_P^2 \,=\, \sum_{j=1}^{k} \sum_{i=1}^{n_j} (y_{ij} - \bar{y}_j)^2\,/\,(n-k)$$

- **Examples:**

```
> y <- c(19, 24, 24, 27, 20, 24, 22, 21, 22, 29, 18, 17)
> f <- factor(rep(1:3, times = 4))
> f

 [1] 1 2 3 1 2 3 1 2 3 1 2 3
Levels: 1 2 3

> n <- 12
> k <- 3
> m.1 <- mean(y[f == "1"])
> m.2 <- mean(y[f == "2"])
> m.3 <- mean(y[f == "3"])
> n1 <- length(y[f == "1"])
> n2 <- length(y[f == "2"])
> n3 <- length(y[f == "3"])
> s2 <- anova(lm(formula = y ~ f))$"Mean Sq"[2]
> s <- sqrt(s2)
> t12 <- (m.2 - m.1)/(s * sqrt(1/n1 + 1/n2))
> t13 <- (m.3 - m.1)/(s * sqrt(1/n3 + 1/n1))
> t23 <- (m.3 - m.2)/(s * sqrt(1/n3 + 1/n2))
> p12 <- 2 * pt(-abs(t12), df = n - k)
> p13 <- 2 * pt(-abs(t13), df = n - k)
> p23 <- 2 * pt(-abs(t23), df = n - k)
> matrix(data = c(p12, p13, NA, p23), dimnames = list(c("2", "3"),
+     c("1", "2")), nrow = 2, ncol = 2)

          1         2
2 0.1981691        NA
3 0.3469732 0.7006709

> pairwise.t.test(y, f, p.adjust.method = "none")
```

```
        Pairwise comparisons using t tests with pooled SD

data:  y and f

  1    2
2 0.20 -
3 0.35 0.70

P value adjustment method: none

> res <- pairwise.t.test(y, f, p.adjust.method = "none")
> res$p.value

          1         2
2 0.1981691        NA
3 0.3469732 0.7006709
```

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()`
- **Input:**

  `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`

$$t = r_{xy} \sqrt{\frac{n-2}{1-r_{xy}^2}} = \frac{\hat{\beta}_2}{s \,/\, \sqrt{ss_x}}$$

$$\text{dove} \quad r_{xy} = \frac{s_{xy}}{s_x \, s_y} = \hat{\beta}_2 \, \frac{s_x}{s_y}$$

  `parameter`

$$df = n - 2$$

  `p.value`

alternative	less	greater	two.sided
p.value	$P(t_{df} \leq t)$	$1 - P(t_{df} \leq t)$	$2\, P(t_{df} \leq -\mid t \mid)$

  `conf.int`

$$\tanh\left(\frac{1}{2} \log\left(\frac{1+r_{xy}}{1-r_{xy}}\right) \mp z_{1-\alpha\,/\,2} \, \frac{1}{\sqrt{n-3}}\right)$$

$$\text{dove} \quad \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{2\,x} - 1}{e^{2\,x} + 1}$$

estimate r_{xy}

- **Example 1:**

```
> x <- c(1, 2, 2, 4, 3, 3)
> y <- c(6, 6, 7, 7, 7, 9)
> n <- 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)
> 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"
```

- **Example 2:**

```
> x <- c(1.2, 1.2, 3.4, 3.4, 4.5, 5.5, 5.5, 5, 6.6, 6.6, 6.6)
> y <- c(1.3, 1.3, 1.3, 4.5, 5.6, 6.7, 6.7, 6.7, 8.8, 8.8, 9)
> n <- 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)
> 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"
```

- **Example 3:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> r <- cov(x, y)/(sd(x) * sd(y))
> r

[1] 0.740661

> t <- r * sqrt((n - 2)/(1 - r^2))
> t

[1] 2.700251

> res <- cor.test(x, y, alternative = "two.sided", conf.level = 0.95)
> res$statistic

       t
2.700251

> parameter <- n - 2
> parameter

[1] 6

> res$parameter

df
 6

> p.value <- 2 * pt(-abs(t), df = n - 2)
> p.value

[1] 0.03556412

> res$p.value

[1] 0.03556412
```

```
> 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] 0.07527696 0.94967566

> res$conf.int

[1] 0.07527696 0.94967566
attr(,"conf.level")
[1] 0.95

> r

[1] 0.740661

> res$estimate

     cor
0.740661

> res$alternative

[1] "two.sided"
```

Test di Kendall

- **Package:** `stats`
- **Sintassi:** `cor.test()`
- **Input:**

  `x` vettore numerico di dimensione n

  `y` vettore numerico di dimensione n

  `alternative = "less" / "greater" / "two.sided"` ipotesi alternativa

  `method = "kendall"`

  `exact = F`

- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value

  `estimate` coefficiente di correlazione campionario

  `alternative` ipotesi alternativa

- **Formula:**

  `statistic`

$$z = \frac{1}{\sigma_K} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{sign}((x_j - x_i)\,(y_j - y_i))$$

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

$$r_{xy}^K = \frac{2\,\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \text{sign}((x_j-x_i)\,(y_j-y_i))}{\left(n\,(n-1)-\sum_{i=1}^{g} t_i\,(t_i-1)\right)^{1\,/\,2}\,\left(n\,(n-1)-\sum_{j=1}^{h} u_j\,(u_j-1)\right)^{1\,/\,2}}$$

- **Example 1:**

```
> x <- c(1, 2, 2, 4, 3, 3)
> y <- c(6, 6, 7, 7, 7, 9)
> n <- 6
> matrice <- matrix(data = 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"
```

- **Example 2:**

```
> x <- c(1.2, 1.2, 3.4, 3.4, 4.5, 5.5, 5.5, 5, 6.6, 6.6, 6.6)
> y <- c(1.3, 1.3, 1.3, 4.5, 5.6, 6.7, 6.7, 6.7, 8.8, 8.8, 9)
> n <- 11
> matrice <- matrix(data = 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"
```

- **Example 3:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> matrice <- matrix(data = 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] 18

> table(x)

x
1.1 2.3 3.4 4.5 5.6 6.7 8.9
  1   1   1   1   1   2   1

> g <- 1
> t1 <- 2
> t <- c(t1)
> t

[1] 2

> table(y)

y
 1.5  6.4  7.8  8.6  8.8 8.86  9.6
   1    1    1    2    1    1    1

> h <- 1
> u1 <- 2
> u <- c(u1)
> u

[1] 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] 7.960468
```

```
> z <- num/sigmaK
> z

[1] 2.261174

> res <- cor.test(x, y, alternative = "two.sided", method = "kendall",
+     exact = F)
> res$statistic

       z
2.261174

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.02374851

> res$p.value

[1] 0.02374851

> cor(x, y, method = "kendall")

[1] 0.6666667

> res$estimate

      tau
0.6666667

> res$alternative

[1] "two.sided"
```

Test Z con una retta di regressione

- **Package:** formularioR
- **Sintassi:** `cor2.test()`
- **Input:**

  `r1` valore di r_{xy}

  `n1` dimensione campionaria n

  `alternative = "less" / "greater" / "two.sided"` ipotesi alternativa

  `rho` valore di ρ_0

  `conf.level` livello di confidenza $1 - \alpha$

- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value

  `conf.int` intervallo di confidenza per il coefficiente di correlazione incognito a livello $1 - \alpha$

  `estimate` coefficiente di correlazione

  `null.value` valore di ρ_0

  `alternative` ipotesi alternativa

- **Formula:**

alternative	less	greater	two.sided
p.value	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\mid z\mid)$

statistic

$$z = \frac{\operatorname{arctanh}(r_{xy}) - \operatorname{arctanh}(\rho_0)}{\frac{1}{\sqrt{n-3}}}$$

$$\text{dove} \quad \operatorname{arctanh}(x) = \frac{1}{2}\,\log\left(\frac{1+x}{1-x}\right)$$

p.value

conf.int

$$\tanh\left(\frac{1}{2}\,\log\left(\frac{1+r_{xy}}{1-r_{xy}}\right) \mp z_{1-\alpha\,/\,2}\,\frac{1}{\sqrt{n-3}}\right)$$

$$\text{dove} \quad \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{2x}-1}{e^{2x}+1}$$

estimate

$$r_{xy}$$

null.value

$$\rho_0$$

- **Example 1:**

```
> x <- c(1.2, 3.4, 5.6, 7.4, 3.2, 5.6, 7.8, 8.9)
> y <- c(1.5, 6.7, 8.5, 4.2, 3.7, 8.8, 9.1, 10.2)
> n <- 8
> r <- cor(x, y)
> r

[1] 0.7354548

> res <- cor2.test(r1 = r, n1 = n, alternative = "two.sided", rho = 0.8,
+     conf.level = 0.95)
> rho0 <- 0.8
> z <- (atanh(r) - atanh(rho0))/(1/sqrt(n - 3))
> z

[1] -0.3535357

> res$statistic

         z
-0.3535357

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.7236869

> res$p.value

[1] 0.7236869

> 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] 0.0638966 0.9485413

> res$conf.int

[1] 0.0638966 0.9485413
attr(,"conf.level")
[1] 0.95

> r

[1] 0.7354548

> res$estimate

        r
0.7354548

> rho0

[1] 0.8

> res$null.value

corr coef
      0.8

> res$alternative

[1] "two.sided"
```

- **Example 2:**

```
> x <- c(1, 2, 2, 4, 3, 3)
> y <- c(6, 6, 7, 7, 7, 9)
> n <- 6
> r <- cor(x, y)
> res <- cor2.test(r1 = r, n1 = n, alternative = "two.sided", rho = 0.6,
+     conf.level = 0.95)
> rho0 <- 0.6
> z <- (atanh(r) - atanh(rho0))/(1/sqrt(n - 3))
> z

[1] -0.1970069

> res$statistic

         z
-0.1970069

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.8438221

> res$p.value

[1] 0.8438221
```

```
> lower <- tanh(atanh(r) - qnorm(1 - 0.05/2)/sqrt(n - 3))
> upper <- tanh(atanh(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

       r
0.522233

> rho0

[1] 0.6

> res$null.value

corr coef
      0.6

> res$alternative

[1] "two.sided"
```

- **Example 3:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> r <- cor(x, y)
> res <- cor2.test(r1 = r, n1 = n, alternative = "two.sided", rho = 0.77,
+     conf.level = 0.95)
> rho0 <- 0.77
> z <- (atanh(r) - atanh(rho0))/(1/sqrt(n - 3))
> z

[1] -0.1529148

> res$statistic

         z
-0.1529148

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.8784655

> res$p.value
```

```
[1] 0.8784655

> lower <- tanh(atanh(r) - qnorm(1 - 0.05/2)/sqrt(n - 3))
> upper <- tanh(atanh(r) + qnorm(1 - 0.05/2)/sqrt(n - 3))
> c(lower, upper)

[1] 0.07527696 0.94967566

> res$conf.int

[1] 0.07527696 0.94967566
attr(,"conf.level")
[1] 0.95

> r

[1] 0.740661

> res$estimate

       r
0.740661

> rho0

[1] 0.77

> res$null.value

corr coef
     0.77

> res$alternative

[1] "two.sided"
```

Test Z con due rette di regressione

- **Package:** formularioR
- **Sintassi:** cor2.test()
- **Input:**

  r1 valore di $r_{x_1 y_1}$

  n1 dimensione campionaria n_1

  r2 valore di $r_{x_2 y_2}$

  n2 dimensione campionaria n_2

  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 i coefficienti di correlazione incogniti a livello $1-\alpha$

`estimate` coefficienti di correlazione

`alternative` ipotesi alternativa

- **Formula:**

`statistic`

$$z = \frac{\text{arctanh}(r_{x_1 y_1}) - \text{arctanh}(r_{x_2 y_2})}{\sqrt{\frac{1}{n_1-3} + \frac{1}{n_2-3}}}$$

$$\text{dove} \quad \text{arctanh}(x) = \frac{1}{2} \log\left(\frac{1+x}{1-x}\right)$$

`p.value`

alternative	less	greater	two.sided
p.value	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\mid z\mid)$

`conf.int`

$$\tanh\left(\frac{1}{2}\log\left(\frac{1+r_{x_1 y_1}}{1-r_{x_1 y_1}}\right) - \frac{1}{2}\log\left(\frac{1+r_{x_2 y_2}}{1-r_{x_2 y_2}}\right) \mp z_{1-\alpha\,/\,2}\sqrt{\frac{1}{n_1-3} + \frac{1}{n_2-3}}\right)$$

$$\text{dove} \quad \tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^{2x}-1}{e^{2x}+1}$$

`estimate`

$$r_{x_1 y_1} \qquad r_{x_2 y_2}$$

- **Example 1:**

```
> x1 <- c(1.2, 3.4, 5.6, 7.4, 3.2, 5.6, 7.8, 8.9)
> y1 <- c(1.5, 6.7, 8.5, 4.2, 3.7, 8.8, 9.1, 10.2)
> n1 <- 8
> r1 <- cor(x1, y1)
> r1

[1] 0.7354548

> x2 <- c(1, 2, 2, 4, 3, 3)
> y2 <- c(6, 6, 7, 7, 7, 9)
> n2 <- 6
> r2 <- cor(x2, y2)
> r2

[1] 0.522233

> res <- cor2.test(r1, n1, r2, n2, alternative = "two.sided", conf.level = 0.95)
> z <- (atanh(r1) - atanh(r2))/sqrt(1/(n1 - 3) + 1/(n2 - 3))
> z

[1] 0.4944581

> res$statistic

        z
0.4944581

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.6209827
```

```
> res$p.value

[1] 0.6209827

> lower <- tanh(atanh(r1) - atanh(r2) - qnorm(1 - 0.05/2) * sqrt(1/(n1 -
+     3) + 1/(n2 - 3)))
> upper <- tanh(atanh(r1) - atanh(r2) + qnorm(1 - 0.05/2) * sqrt(1/(n1 -
+     3) + 1/(n2 - 3)))
> c(lower, upper)

[1] -0.7895570  0.9460192

> res$conf.int

[1] -0.7895570  0.9460192
attr(,"conf.level")
[1] 0.95

> c(r1, r2)

[1] 0.7354548 0.5222330

> res$estimate

       r1        r2
0.7354548 0.5222330

> res$alternative

[1] "two.sided"
```

- **Example 2:**

```
> x1 <- c(1.2, 5.6, 7.4, 6.78, 6.3, 7.8, 8.9)
> y1 <- c(2.4, 6.4, 8.4, 8.5, 8.54, 8.7, 9.7)
> n1 <- 7
> r1 <- cor(x1, y1)
> r1

[1] 0.9755886

> x2 <- c(3.7, 8.6, 9.9, 10.4)
> y2 <- c(5.8, 9.7, 12.4, 15.8)
> n2 <- 4
> r2 <- cor(x2, y2)
> r2

[1] 0.9211733

> res <- cor2.test(r1, n1, r2, n2, alternative = "two.sided", conf.level = 0.95)
> z <- (atanh(r1) - atanh(r2))/sqrt(1/(n1 - 3) + 1/(n2 - 3))
> z

[1] 0.5367157

> res$statistic

        z
0.5367157
```

```
> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.591464

> res$p.value

[1] 0.591464

> lower <- tanh(atanh(r1) - atanh(r2) - qnorm(1 - 0.05/2) * sqrt(1/(n1 -
+     3) + 1/(n2 - 3)))
> upper <- tanh(atanh(r1) - atanh(r2) + qnorm(1 - 0.05/2) * sqrt(1/(n1 -
+     3) + 1/(n2 - 3)))
> c(lower, upper)

[1] -0.9203392  0.9925038

> res$conf.int

[1] -0.9203392  0.9925038
attr(,"conf.level")
[1] 0.95

> c(r1, r2)

[1] 0.9755886 0.9211733

> res$estimate

       r1        r2
0.9755886 0.9211733

> res$alternative

[1] "two.sided"
```

- **Example 3:**

```
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y1 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> n1 <- 8
> r1 <- cor(x1, y1)
> r1

[1] 0.8260355

> x2 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y2 <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n2 <- 8
> r2 <- cor(x2, y2)
> r2

[1] 0.8531061

> res <- cor2.test(r1, n1, r2, n2, alternative = "two.sided", conf.level = 0.95)
> z <- (atanh(r1) - atanh(r2))/sqrt(1/(n1 - 3) + 1/(n2 - 3))
> z

[1] -0.1453518
```

```
> res$statistic

         z
-0.1453518

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.8844331

> res$p.value

[1] 0.8844331

> lower <- tanh(atanh(r1) - atanh(r2) - qnorm(1 - 0.05/2) * sqrt(1/(n1 -
+     3) + 1/(n2 - 3)))
> upper <- tanh(atanh(r1) - atanh(r2) + qnorm(1 - 0.05/2) * sqrt(1/(n1 -
+     3) + 1/(n2 - 3)))
> c(lower, upper)

[1] -0.8696200  0.8169779

> res$conf.int

[1] -0.8696200  0.8169779
attr(,"conf.level")
[1] 0.95

> c(r1, r2)

[1] 0.8260355 0.8531061

> res$estimate

       r1        r2
0.8260355 0.8531061

> res$alternative

[1] "two.sided"
```

9.2 Test di ipotesi sulla autocorrelazione

Test di Box - Pierce

- **Package:** stats
- **Sintassi:** `Box.test()`
- **Input:**

  `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

$$df = d$$

p.value

$$P(\chi^2_{df} \geq c)$$

- **Example 1:**

```
> x <- c(1.2, 3.4, 5.6, 7.4, 3.2, 5.6, 7.8, 8.9)
> n <- 8
> d <- 3
> autocorr <- as.vector(acf(x, lag.max = d, plot = F)[[1]])
> autocorr <- autocorr[-1]
> autocorr

[1]  0.2562830 -0.1947304 -0.1413042

> c <- n * sum(autocorr^2)
> c

[1] 0.9885422

> Box.test(x, lag = d)$statistic

X-squared
0.9885422

> d

[1] 3

> Box.test(x, lag = d)$parameter

df
 3

> p.value <- 1 - pchisq(c, df = d)
> p.value

[1] 0.8040244

> Box.test(x, lag = d)$p.value

[1] 0.8040244
```

- **Example 2:**

```
> x <- c(1.2, 2.6, 3.8, 4.4, 5.2)
> n <- 5
> d <- 2
> autocorr <- as.vector(acf(x, lag.max = d, plot = F)[[1]])
> autocorr <- autocorr[-1]
> autocorr

[1]  0.36612642 -0.09918963

> c <- n * sum(autocorr^2)
> c

[1] 0.7194357

> Box.test(x, lag = d)$statistic

X-squared
0.7194357

> d

[1] 2

> Box.test(x, lag = d)$parameter

df
 2

> p.value <- 1 - pchisq(c, df = d)
> p.value

[1] 0.6978732

> Box.test(x, lag = d)$p.value

[1] 0.6978732
```

- **Example 3:**

```
> x <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> n <- 8
> d <- 2
> autocorr <- as.vector(acf(x, lag.max = d, plot = F)[[1]])
> autocorr <- autocorr[-1]
> autocorr

[1]  0.2271066 -0.2233210

> c <- n * sum(autocorr^2)
> c

[1] 0.8115975

> Box.test(x, lag = d)$statistic

X-squared
0.8115975

> d
```

```
[1] 2

> Box.test(x, lag = d)$parameter

df
 2

> p.value <- 1 - pchisq(c, df = d)
> p.value

[1] 0.6664443

> Box.test(x, lag = d)$p.value

[1] 0.6664443
```

Test di Ljung - Box

- **Package:** stats
- **Sintassi:** Box.test()
- **Input:**

  x vettore numerico di dimensione n

  lag il valore d del ritardo

  type = "Ljung-Box"
- **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

$$df = d$$

  p.value

$$P(\chi^2_{df} \geq c)$$

- **Example 1:**

```
> x <- c(1.2, 3.4, 5.6, 7.4, 3.2, 5.6, 7.8, 8.9)
> n <- 8
> d <- 3
> autocorr <- as.vector(acf(x, lag.max = d, plot = F)[[1]])
> autocorr <- autocorr[-1]
> autocorr

[1]  0.2562830 -0.1947304 -0.1413042
```

```
> c <- n * (n + 2) * sum(autocorr^2/(n - 1:d))
> c

[1] 1.575709

> Box.test(x, lag = d, type = "Ljung-Box")$statistic

X-squared
 1.575709

> d

[1] 3

> Box.test(x, lag = d, type = "Ljung-Box")$parameter

df
 3

> p.value <- 1 - pchisq(c, df = d)
> p.value

[1] 0.6649102

> Box.test(x, lag = d, type = "Ljung-Box")$p.value

[1] 0.6649102
```

- **Example 2:**

```
> x <- c(1.2, 2.6, 3.8, 4.4, 5.2)
> n <- 5
> d <- 2
> autocorr <- as.vector(acf(x, lag.max = d, plot = F)[[1]])
> autocorr <- autocorr[-1]
> autocorr

[1]  0.36612642 -0.09918963

> c <- n * (n + 2) * sum(autocorr^2/(n - 1:d))
> c

[1] 1.287708

> Box.test(x, lag = d, type = "Ljung-Box")$statistic

X-squared
 1.287708

> d

[1] 2

> Box.test(x, lag = d, type = "Ljung-Box")$parameter

df
 2
```

```
> p.value <- 1 - pchisq(c, df = d)
> p.value

[1] 0.5252641

> Box.test(x, lag = d, type = "Ljung-Box")$p.value

[1] 0.5252641
```

- **Example 3:**

```
> x <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> n <- 8
> d <- 2
> autocorr <- as.vector(acf(x, lag.max = d, plot = F)[[1]])
> autocorr <- autocorr[-1]
> autocorr

[1]  0.2271066 -0.2233210

> c <- n * (n + 2) * sum(autocorr^2/(n - 1:d))
> c

[1] 1.254420

> Box.test(x, lag = d, type = "Ljung-Box")$statistic

X-squared
 1.254420

> d

[1] 2

> Box.test(x, lag = d, type = "Ljung-Box")$parameter

df
 2

> p.value <- 1 - pchisq(c, df = d)
> p.value

[1] 0.5340799

> Box.test(x, lag = d, type = "Ljung-Box")$p.value

[1] 0.5340799
```

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: $R_j \quad \forall\, j = 1, 2, \ldots, k$
- media dei ranghi nel campione j-esimo: $\bar{R}_j \quad \forall\, j = 1, 2, \ldots, k$
- media dei ranghi nel campione di dimensione n: $\bar{R}$
- ties nel campione di dimensione n: $t_j \quad \forall\, j = 1, 2, \ldots, g \qquad \sum_{j=1}^{g} t_j = n \qquad 1 \leq g \leq n$

10.2 Test di ipotesi sulla mediana con uno o due campioni

Test esatto Wilcoxon signed rank

- **Package:** `stats`
- **Sintassi:** `wilcox.test()`
- **Input:**

  `x` vettore numerico di dimensione n

  `mu` il valore di $Q_{0.5}(x)_{|\, H_0}$

  `alternative = "less" / "greater" / "two.sided"` ipotesi alternativa

  `exact = TRUE`
- **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

$$Q_{0.5}(x)_{| H_0}$$

- **Example 1:**

```
> x <- c(-0.1, -0.2, 0.7, 0.8, -1.2, -1.6, 2, 3.4, 3.7)
> n <- 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

> 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

> res1 <- wilcox.test(x, mu = 3.3, alternative = "less", exact = TRUE)
> res1$statistic

V
3

> p.value.less <- psignrank(v, n)
> p.value.less

[1] 0.009765625

> res1$p.value

[1] 0.009765625

> p.value.greater <- 1 - psignrank(v - 1, n)
> p.value.greater

[1] 0.9941406

> res2 <- wilcox.test(x, mu = 3.3, alternative = "greater", exact = TRUE)
> res2$p.value

[1] 0.9941406

> p.value.two.sided <- 2 * min(p.value.less, p.value.greater)
> p.value.two.sided

[1] 0.01953125

> res3 <- wilcox.test(x, mu = 3.3, alternative = "two.sided", exact = TRUE)
> res3$p.value

[1] 0.01953125
```

- **Example 2:**

```
> x <- c(3.8, 5.6, 1.8, 5, 2.4, 4.2, 7.3, 8.6, 9.1, 5.2)
> n <- 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

> 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

> res1 <- wilcox.test(x, mu = 6.3, alternative = "less", exact = TRUE)
> res1$statistic

 V
16

> p.value.less <- psignrank(v, n)
> p.value.less

[1] 0.1376953

> res1$p.value

[1] 0.1376953

> p.value.greater <- 1 - psignrank(v - 1, n)
> p.value.greater

[1] 0.883789

> res2 <- wilcox.test(x, mu = 6.3, alternative = "greater", exact = TRUE)
> res2$p.value

[1] 0.883789

> p.value.two.sided <- 2 * min(p.value.less, p.value.greater)
> p.value.two.sided

[1] 0.2753906

> res3 <- wilcox.test(x, mu = 6.3, alternative = "two.sided", exact = TRUE)
> res3$p.value

[1] 0.2753906
```

- **Example 3:**

```
> x <- c(1.2, 3.4, 4.5, 6.4, 3, 4, 2.3, 8.8, 9.87, 12.34)
> n <- 10
> mu <- 2.7
> xx <- rank(abs(x - mu)) * sign(x - mu)
> xx
```

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

```
> v <- sum(xx[xx > 0])
> v
```

```
[1] 48
```

```
> res1 <- wilcox.test(x, mu = 2.7, alternative = "less", exact = TRUE)
> res1$statistic
```

```
 V
48
```

```
> p.value.less <- psignrank(v, n)
> p.value.less
```

```
[1] 0.9863281
```

```
> res1$p.value
```

```
[1] 0.9863281
```

```
> p.value.greater <- 1 - psignrank(v - 1, n)
> p.value.greater
```

```
[1] 0.01855469
```

```
> res2 <- wilcox.test(x, mu = 2.7, alternative = "greater", exact = TRUE)
> res2$p.value
```

```
[1] 0.01855469
```

```
> p.value.twosided <- 2 * min(p.value.less, p.value.greater)
> p.value.twosided
```

```
[1] 0.03710938
```

```
> res3 <- wilcox.test(x, mu = 2.7, alternative = "two.sided", exact = TRUE)
> res3$p.value
```

```
[1] 0.03710938
```

- **Note:** Il vettore `abs(x-mu)` non deve contenere valori duplicati o nulli.

Test asintotico Wilcoxon signed rank

- **Package:** stats
- **Sintassi:** wilcox.test()
- **Input:**

  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 = TRUE / FALSE correzione di continuità di *Yates*

  exact = FALSE
- **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

alternative	less	greater	two.sided
p.value	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\mid z\mid)$

correct = TRUE

$$z = \frac{v - \frac{m\,(m+1)}{4} + 0.5}{\left[\frac{1}{24}\left(m\,(m+1)\,(2\,m+1) - \frac{1}{2}\sum_{j=1}^{g} t_j\,(t_j^2 - 1)\right)\right]^{1/2}}$$

correct = FALSE

$$z = \frac{v - \frac{m\,(m+1)}{4}}{\left[\frac{1}{24}\left(m\,(m+1)\,(2\,m+1) - \frac{1}{2}\sum_{j=1}^{g} t_j\,(t_j^2 - 1)\right)\right]^{1/2}}$$

  null.value

$$Q_{0.5}(x)_{\mid H_0}$$

- **Example 1:**

```
> x <- c(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 <- 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   5.5   5.5

> v <- sum(xx[xx > 0])
> v

[1] 38.5

> res <- wilcox.test(x, mu = 4, alternative = "less", correct = FALSE,
+     exact = FALSE)
> res$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

> res$p.value

[1] 0.4832509
```

- **Example 2:**

```
> x <- c(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 <- 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  7.5  7.5
[16] -7.5  7.5 18.0 18.0  7.5  7.5

> v <- sum(xx[xx > 0])
> v
```

```
[1] 216

> res <- wilcox.test(x, mu = 3, alternative = "less", correct = TRUE,
+     exact = FALSE)
> res$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

> res$p.value

[1] 0.999871
```

- **Example 3:**

```
> x <- c(1.2, 3.4, 4.5, 6.4, 3, 4, 2.3, 8.8, 9.87, 12.34)
> n <- 10
> mu <- 2.7
> xx <- (x - mu)[(x - mu) != 0]
> xx <- c(-1.5, 0.7, 1.8, 3.7, 0.3, 1.3, -0.4, 6.1, 7.17, 9.64)
> m <- length(xx)
> m

[1] 10

> xx <- rank(abs(xx)) * sign(xx)
> xx

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

> v <- sum(xx[xx > 0])
> v

[1] 48

> res <- wilcox.test(x, mu = 2.7, alternative = "less", correct = TRUE,
+     exact = FALSE)
> res$statistic

 V
48
```

```
> table(rank(abs(xx)))

 1  2  3  4  5  6  7  8  9 10
 1  1  1  1  1  1  1  1  1  1

> g <- 10
> t1 <- 1
> t2 <- 1
> t3 <- 1
> t4 <- 1
> t5 <- 1
> t6 <- 1
> t7 <- 1
> t8 <- 1
> t9 <- 1
> t10 <- 1
> t <- c(t1, t2, t3, t4, t5, t6, t7, t8, t9, t10)
> 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] 0.9838435

> res$p.value

[1] 0.9838435
```

Test esatto di Mann - Whitney

- **Package:** `stats`
- **Sintassi:** `wilcox.test()`
- **Input:**

  `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

  `exact = TRUE`

- **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	$P(W \leq w)$	$P(W \geq w)$	$2 \min\left(P(W \leq w),\, P(W \geq w)\right)$

null.value

$$\left(\, Q_{0.5}(x) - Q_{0.5}(y)\,\right)_{\mid\, H_0}$$

- **Example 1:**

```
> x <- c(1.2, 3.4, 5.4, -5.6, 7.3, 2.1)
> nx <- 6
> y <- c(-1.1, -0.1, 0.9, 1.9, 2.9, 3.9, 4.99)
> ny <- 7
> mu <- -2.1
> c(x, y + mu)

 [1]  1.20  3.40  5.40 -5.60  7.30  2.10 -3.20 -2.20 -1.20 -0.20  0.80  1.80
[13]  2.89

> Rx <- sum(rank(c(x, y + mu))[1:nx])
> Rx

[1] 53

> w <- Rx - nx * (nx + 1)/2
> w

[1] 32

> res1 <- wilcox.test(x, y, mu = -2.1, alternative = "less", exact = TRUE)
> res1$statistic

 W
32

> p.value.less <- pwilcox(w, nx, ny)
> p.value.less

[1] 0.9493007

> res1$p.value

[1] 0.9493007

> p.value.greater <- 1 - pwilcox(w - 1, nx, ny)
> p.value.greater

[1] 0.06876457

> res2 <- wilcox.test(x, y, mu = -2.1, alternative = "greater",
+     exact = TRUE)
> res2$p.value

[1] 0.06876457

> p.value.two.sided <- 2 * min(p.value.less, p.value.greater)
> p.value.two.sided
```

```
[1] 0.1375291

> res3 <- wilcox.test(x, y, mu = -2.1, alternative = "two.sided",
+     exact = TRUE)
> res3$p.value

[1] 0.1375291
```

- **Example 2:**

```
> x <- c(33.3, 30.1, 38.62, 38.94, 42.63, 41.96, 46.3, 43.25)
> nx <- 8
> y <- c(31.62, 46.33, 31.82, 40.21, 45.72, 39.8, 45.6, 41.25)
> ny <- 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 32.92 41.31
[13] 46.82 40.90 46.70 42.35

> Rx <- sum(rank(c(x, y + mu))[1:nx])
> Rx

[1] 61

> w <- Rx - nx * (nx + 1)/2
> w

[1] 25

> res1 <- wilcox.test(x, y, mu = 1.1, alternative = "less", exact = TRUE)
> res1$statistic

 W
25

> p.value.less <- pwilcox(w, nx, ny)
> p.value.less

[1] 0.2526807

> res1$p.value

[1] 0.2526807

> p.value.greater <- 1 - pwilcox(w - 1, nx, ny)
> p.value.greater

[1] 0.7790987

> res2 <- wilcox.test(x, y, mu = 1.1, alternative = "greater",
+     exact = TRUE)
> res2$p.value

[1] 0.7790987

> p.value.two.sided <- 2 * min(p.value.less, p.value.greater)
> p.value.two.sided

[1] 0.5053613

> res3 <- wilcox.test(x, y, mu = 1.1, alternative = "two.sided",
+     exact = TRUE)
> res3$p.value

[1] 0.5053613
```

- **Example 3:**

```
> x <- c(4, 2.3, 8.8, 9.87, 12.34, 1.4)
> nx <- 6
> y <- c(6.4, 9.6, 8.86, 7.8, 8.6, 8.7, 1.1)
> ny <- 7
> mu <- 2.3
> c(x, y + mu)

 [1]  4.00  2.30  8.80  9.87 12.34  1.40  8.70 11.90 11.16 10.10 10.90 11.00
[13]  3.40

> Rx <- sum(rank(c(x, y + mu))[1:nx])
> Rx

[1] 33

> w <- Rx - nx * (nx + 1)/2
> w

[1] 12

> res1 <- wilcox.test(x, y, mu = 2.3, alternative = "less", exact = TRUE)
> res1$statistic

 W
12

> p.value.less <- pwilcox(w, nx, ny)
> p.value.less

[1] 0.1171329

> res1$p.value

[1] 0.1171329

> p.value.greater <- 1 - pwilcox(w - 1, nx, ny)
> p.value.greater

[1] 0.9096737

> res2 <- wilcox.test(x, y, mu = 2.3, alternative = "greater",
+     exact = TRUE)
> res2$p.value

[1] 0.9096737

> p.value.two.sided <- 2 * min(p.value.less, p.value.greater)
> p.value.two.sided

[1] 0.2342657

> res3 <- wilcox.test(x, y, mu = 2.3, alternative = "two.sided",
+     exact = TRUE)
> res3$p.value

[1] 0.2342657
```

- **Note:** Il vettore `c(x, y+mu)` non deve contenere valori duplicati.

Test asintotico di Mann - Whitney

- **Package:** stats
- **Sintassi:** wilcox.test()
- **Input:**

  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 = TRUE / FALSE correzione di continuità di *Yates*

  exact = FALSE

- **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	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\,|\,z\,|)$

correct = TRUE

$$z = \frac{w - \frac{n_x\,n_y}{2} + 0.5}{\left[\frac{n_x\,n_y}{12}\left(n_x + n_y + 1 - \frac{\sum_{j=1}^{g} t_j\,(t_j^2 - 1)}{(n_x + n_y)\,(n_x + n_y - 1)}\right)\right]^{1\,/\,2}}$$

correct = FALSE

$$z = \frac{w - \frac{n_x\,n_y}{2}}{\left[\frac{n_x\,n_y}{12}\left(n_x + n_y + 1 - \frac{\sum_{j=1}^{g} t_j\,(t_j^2 - 1)}{(n_x + n_y)\,(n_x + n_y - 1)}\right)\right]^{1\,/\,2}}$$

  null.value

$$(\,Q_{0.5}(x) - Q_{0.5}(y)\,)_{|\,H_0}$$

- **Example 1:**

```
> x <- c(-1, 1, -2, -1, 1, 1, 1, 1, -1, -2, 1, 1)
> nx <- 12
> y <- c(1, 1, 2, 3, 4, 5, 3, 2, 1)
> ny <- 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
```

```
> res <- wilcox.test(x, y, mu = -4, alternative = "less", correct = TRUE,
+     exact = FALSE)
> res$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

> res$p.value

[1] 0.9910242
```

- **Example 2:**

```
> x <- c(33.3, 30.1, 38.62, 38.94, 42.63, 41.96, 46.3, 43.25)
> nx <- 8
> y <- c(31.62, 46.33, 31.82, 40.21, 45.72, 39.8, 45.6, 41.25)
> ny <- 8
> mu <- 4
> Rx <- sum(rank(c(x, y + mu))[1:nx])
> Rx

[1] 51

> w <- Rx - nx * (nx + 1)/2
> w

[1] 15

> res <- wilcox.test(x, y, mu = 4, alternative = "less", correct = FALSE,
+     exact = FALSE)
> res$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
```

```
> res$p.value
```

```
[1] 0.03710171
```

- **Example 3:**

```
> x <- c(4, 2.3, 8.8, 9.87, 12.34, 1.4)
> nx <- 6
> y <- c(6.4, 9.6, 8.86, 7.8, 8.6, 8.7, 1.1)
> ny <- 7
> mu <- 2.3
> Rx <- sum(rank(c(x, y + mu))[1:nx])
> Rx

[1] 33

> w <- Rx - nx * (nx + 1)/2
> w

[1] 12

> res <- wilcox.test(x, y, mu = 2.3, alternative = "less", correct = TRUE,
+     exact = FALSE)
> res$statistic

 W
12

> table(rank(c(x, y + mu)))

 1  2  3  4  5  6  7  8  9 10 11 12 13
 1  1  1  1  1  1  1  1  1  1  1  1  1

> g <- 13
> t1 <- 1
> t2 <- 1
> t3 <- 1
> t4 <- 1
> t5 <- 1
> t6 <- 1
> t7 <- 1
> t8 <- 1
> t9 <- 1
> t10 <- 1
> t11 <- 1
> t12 <- 1
> t13 <- 1
> t <- c(t1, t2, t3, t4, t5, t6, t7, t8, t9, t10, t11, t12, t13)
> 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] 0.1123193

> res$p.value

[1] 0.1123193
```

Test esatto Wilcoxon signed rank per dati appaiati

- **Package:** stats
- **Sintassi:** wilcox.test()
- **Input:**

  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

  exact = TRUE

  paired = TRUE

- **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

  $$(Q_{0.5}(x) - Q_{0.5}(y))_{|H_0}$$

- **Example 1:**

```
> x <- c(-0.1, -0.2, 0.7, 0.8, -1.2, -1.6, 2, 3.4, 3.7)
> n <- 9
> y <- c(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

> 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

> res1 <- wilcox.test(x, y, mu = -4, alternative = "less", exact = TRUE,
+     paired = TRUE)
> res1$statistic

 V
21
```

```
> p.value.less <- psignrank(v, n)
> p.value.less

[1] 0.4550781

> res1$p.value

[1] 0.4550781

> p.value.greater <- 1 - psignrank(v - 1, n)
> p.value.greater

[1] 0.5898438

> res2 <- wilcox.test(x, y, mu = -4, alternative = "greater", paired = TRUE,
+     exact = TRUE)
> res2$p.value

[1] 0.5898438

> p.value.two.sided <- 2 * min(p.value.less, p.value.greater)
> p.value.two.sided

[1] 0.9101562

> res3 <- wilcox.test(x, y, mu = -4, alternative = "two.sided",
+     paired = TRUE, exact = TRUE)
> res3$p.value

[1] 0.9101562
```

- **Example 2:**

```
> x <- c(33.3, 30.1, 38.62, 38.94, 42.63, 41.96, 46.3, 43.25)
> n <- 8
> y <- c(31.62, 46.33, 31.82, 40.21, 45.72, 39.8, 45.6, 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

> 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

> res1 <- wilcox.test(x, y, mu = 1.1, alternative = "less", exact = TRUE,
+     paired = TRUE)
> res1$statistic

 V
16
```

```
> p.value.less <- psignrank(v, n)
> p.value.less

[1] 0.421875

> res1$p.value

[1] 0.421875

> p.value.greater <- 1 - psignrank(v - 1, n)
> p.value.greater

[1] 0.6289062

> res2 <- wilcox.test(x, y, mu = 1.1, alternative = "greater",
+     exact = TRUE, paired = TRUE)
> res2$p.value

[1] 0.6289062

> p.value.two.sided <- 2 * min(p.value.less, p.value.greater)
> p.value.two.sided

[1] 0.84375

> res3 <- wilcox.test(x, y, mu = 1.1, alternative = "two.sided",
+     exact = TRUE, paired = TRUE)
> res3$p.value

[1] 0.84375
```

- **Example 3:**

```
> x <- c(4, 2.3, 8.8, 9.87, 12.34, 1.4)
> n <- 6
> y <- c(6.4, 9.6, 8.86, 7.8, 8.6, 8.8)
> mu <- 2.3
> x - y - mu

[1] -4.70 -9.60 -2.36 -0.23  1.44 -9.70

> xy <- rank(abs(x - y - mu)) * sign(x - y - mu)
> xy

[1] -4 -5 -3 -1  2 -6

> v <- sum(xy[xy > 0])
> v

[1] 2

> res1 <- wilcox.test(x, y, mu = 2.3, alternative = "less", exact = TRUE,
+     paired = TRUE)
> res1$statistic

V
2
```

```
> p.value.less <- psignrank(v, n)
> p.value.less

[1] 0.046875

> res2 <- wilcox.test(x, y, mu = 2.3, alternative = "less", exact = TRUE,
+     paired = TRUE)
> res2$p.value

[1] 0.046875

> p.value.greater <- 1 - psignrank(v - 1, n)
> p.value.greater

[1] 0.96875

> res2$p.value

[1] 0.046875

> p.value.two.sided <- 2 * min(p.value.less, p.value.greater)
> p.value.two.sided

[1] 0.09375

> res3 <- wilcox.test(x, y, mu = 2.3, alternative = "two.sided",
+     exact = TRUE, paired = TRUE)
> res3$p.value

[1] 0.09375
```

- **Note:** 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()`
- **Input:**

  `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 = TRUE / FALSE` correzione di continuità di *Yates*

  `exact = FALSE`

  `paired = TRUE`

- **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:**

alternative	less	greater	two.sided
p.value	$\Phi(z)$	$1-\Phi(z)$	$2\,\Phi(-\mid z\mid)$

statistic

$$v$$

p.value

correct = TRUE

$$z = \frac{v - \frac{m\,(m+1)}{4} + 0.5}{\left[\frac{1}{24}\left(m\,(m+1)\,(2\,m+1) - \frac{1}{2}\,\sum_{j=1}^{g} t_j\,(t_j^2-1)\right)\right]^{1/2}}$$

correct = FALSE

$$z = \frac{v - \frac{m\,(m+1)}{4}}{\left[\frac{1}{24}\left(m\,(m+1)\,(2\,m+1) - \frac{1}{2}\,\sum_{j=1}^{g} t_j\,(t_j^2-1)\right)\right]^{1/2}}$$

null.value

$$\left(\,Q_{0.5}(x) - Q_{0.5}(y)\,\right)_{|\,H_0}$$

- **Example 1:**

```
> x <- c(4, 4, 3, 4, 2, 4, 5, 5, 4, 3.3)
> n <- 10
> y <- c(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

> res <- wilcox.test(x, y, mu = -2, alternative = "less", correct = TRUE,
+     exact = FALSE, paired = TRUE)
> res$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

> res$p.value

[1] 0.6883942
```

- **Example 2:**

```
> x <- c(33.3, 30.1, 38.62, 38.94, 42.63, 41.96, 46.3, 43.25)
> n <- 8
> y <- c(31.62, 46.33, 31.82, 40.21, 45.72, 39.8, 45.6, 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

> res <- wilcox.test(x, y, mu = 2, alternative = "less", correct = FALSE,
+     exact = FALSE, paired = TRUE)
> res$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

> res$p.value

[1] 0.08814819
```

- **Example 3:**

```
> x <- c(4.5, 6.4, 3, 4, 2.3, 8.8, 9.87, 12.34)
> n <- 8
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> mu <- 2.3
> xy <- (x - y - mu)[(x - y - mu) != 0]
> xy

[1]  0.70 -2.30 -8.90 -7.10 -8.86 -1.30 -1.03  1.44

> m <- length(xy)
> m

[1] 8

> xy <- rank(abs(xy)) * sign(xy)
> xy

[1]  1 -5 -8 -6 -7 -3 -2  4

> v <- sum(xy[xy > 0])
> v

[1] 5

> res <- wilcox.test(x, y, mu = 2.3, alternative = "less", correct = TRUE,
+     exact = FALSE, paired = TRUE)
> res$statistic

V
5

> table(rank(abs(xy)))

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 <- 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] 0.04002896

> res$p.value

[1] 0.04002896
```

10.3 Test di ipotesi sulla mediana con più campioni

Test di Kruskal - Wallis

- **Package:** stats
- **Sintassi:** kruskal.test()
- **Input:**

  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

$$c = \frac{1}{C} \frac{12}{n\,(n+1)} \sum_{i=1}^{k} n_i \left(\bar{R}_i - \bar{R}\right)^2 = \frac{1}{C} \frac{12}{n\,(n+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3\,(n+1)$$

$$\text{dove} \quad C = 1 - \frac{\sum_{i=1}^{h} t_i\,(t_i^2 - 1)}{n\,(n^2 - 1)} \quad \text{e} \quad \bar{R} = \frac{1}{n} \sum_{i=1}^{k} R_i = \frac{1}{n} \sum_{i=1}^{k} n_i\,\bar{R}_i = \frac{n+1}{2}$$

  parameter

$$df = k - 1$$

  p.value

$$P(\chi^2_{df} \geq c)$$

- **Example 1:**

```
> x <- c(2.1, 3, 2.1, 5.3, 5.3, 2.1, 5.6, 7.5, 2.1, 5.3, 2.1, 7.5)
> g <- factor(rep(letters[1:4], each = 3))
> g
```

```
 [1] a a a b b b c c c d d d
Levels: a b c d

> n <- 12
> k <- 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)
> tapply(x, g, FUN = "length")

a b c d
3 3 3 3

> n1 <- 3
> n2 <- 3
> n3 <- 3
> n4 <- 3
> enne <- c(n1, n2, n3, n4)
> C <- 1 - sum(t * (t^2 - 1))/(n * (n^2 - 1))
> statistic <- (12/(n * (n + 1)) * sum(R^2/enne) - 3 * (n + 1))/C
> statistic

[1] 2.542784

> res <- kruskal.test(x, g)
> res$statistic

Kruskal-Wallis chi-squared
                  2.542784

> parameter <- k - 1
> parameter

[1] 3

> res$parameter

df
 3

> p.value <- 1 - pchisq(statistic, df = parameter)
> p.value

[1] 0.4676086
```

```
> res$p.value

[1] 0.4676086
```

- **Example 2:**

```
> x <- c(0.7, 1.6, 0.2, 1.2, 0.1, 3.4, 3.7, 0.8, 0, 2, 1.9, 0.8,
+     1.1, 0.1, 0.1, 4.4, 5.5, 1.6, 4.6, 3.4)
> g <- factor(rep(letters[1:2], each = 10))
> g

 [1] a a a a a a a a a a b b b b b b b b b b
Levels: a b

> n <- 20
> k <- 2
> R1 <- sum(rank(x)[g == "a"])
> R2 <- sum(rank(x)[g == "b"])
> R <- c(R1, R2)
> R

[1]  90.5 119.5

> table(rank(x))

   1    3    5    6  7.5    9   10 11.5   13   14 15.5   17   18   19   20
   1    3    1    1    2    1    1    2    1    1    2    1    1    1    1

> h <- 4
> t1 <- 3
> t2 <- 2
> t3 <- 2
> t4 <- 2
> t <- c(t1, t2, t3, t4)
> tapply(x, g, FUN = "length")

 a  b
10 10

> n1 <- 10
> n2 <- 10
> enne <- c(n1, n2)
> C <- 1 - sum(t * (t^2 - 1))/(n * (n^2 - 1))
> statistic <- (12/(n * (n + 1)) * sum(R^2/enne) - 3 * (n + 1))/C
> statistic

[1] 1.207785

> res <- kruskal.test(x, g)
> res$statistic

Kruskal-Wallis chi-squared
                  1.207785

> parameter <- k - 1
> parameter

[1] 1

> res$parameter
```

```
df
 1

> p.value <- 1 - pchisq(statistic, df = parameter)
> p.value

[1] 0.2717712

> res$p.value

[1] 0.2717712
```

- **Example 3:**

```
> x <- c(4, 2.3, 8.8, 9.87, 12.34, 1.4, 6.4, 9.6, 8.86, 7.8, 8.6,
+     8.8, 2, 0.3)
> g <- factor(rep(c("Ctl", "Trt"), times = c(10, 4)))
> g

 [1] Ctl Ctl Ctl Ctl Ctl Ctl Ctl Ctl Ctl Ctl Trt Trt Trt Trt
Levels: Ctl Trt

> n <- 14
> k <- 2
> R1 <- sum(rank(x)[g == "Ctl"])
> R2 <- sum(rank(x)[g == "Trt"])
> R <- c(R1, R2)
> R

[1] 83.5 21.5

> table(rank(x))

  1   2   3   4   5   6   7   8 9.5  11  12  13  14
  1   1   1   1   1   1   1   1   2   1   1   1   1

> h <- 1
> t1 <- 2
> t <- c(t1)
> tapply(x, g, FUN = "length")

Ctl Trt
 10   4

> n1 <- 10
> n2 <- 4
> enne <- c(n1, n2)
> C <- 1 - sum(t * (t^2 - 1))/(n * (n^2 - 1))
> statistic <- (12/(n * (n + 1)) * sum(R^2/enne) - 3 * (n + 1))/C
> statistic

[1] 1.448183

> res <- kruskal.test(x, g)
> res$statistic

Kruskal-Wallis chi-squared
                  1.448183
```

```
> parameter <- k - 1
> parameter

[1] 1

> res$parameter

df
 1

> p.value <- 1 - pchisq(statistic, df = parameter)
> p.value

[1] 0.2288198

> res$p.value

[1] 0.2288198
```

10.4 Test di ipotesi sull'omogeneità delle varianze

Test di Levene

- **Package:** car
- **Sintassi:** `levene.test()`
- **Input:**

  `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`

$$Fvalue = \frac{\left[\sum_{j=1}^{k}\sum_{i=1}^{n_j}(xij - \bar{x}_j)^2\right] / (k-1)}{\left[\sum_{j=1}^{k}(n_j - 1)\, s_j^2\right] \,/\, (n-k)}$$

$$\text{dove} \quad x_{ij} = \left|y_{ij} - Q_{0.5}\left(\left\{y_{1j}, \ldots, y_{n_j j}\right\}\right)\right| \quad \forall\, j = 1, 2, \ldots, k \quad \forall\, i = 1, 2, \ldots, n_j$$

  `Pr(>F)`

$$P(F_{k-1,\, n-k} \geq Fvalue)$$

- **Example 1:**

```
> y <- c(1, 4, 10, 2.1, 3.5, 5.6, 8.4, 12, 16.5, 22, 1.2, 3.4)
> f <- factor(rep(letters[1:4], each = 3))
> n <- 12
> k <- 4
> Df <- c(k - 1, n - k)
> Df

[1] 3 8

> res <- levene.test(y, group = f)
> res$Df

[1] 3 8

> x <- abs(y - ave(y, f, FUN = "median"))
> Fvalue <- anova(lm(formula = x ~ f))$F
> Fvalue

[1] 0.608269       NA

> res$"F value"

[1] 0.608269       NA

> p.value <- 1 - pf(Fvalue, df1 = k - 1, df2 = n - k)
> p.value

[1] 0.6281414        NA

> res$"Pr(>F)"

[1] 0.6281414        NA
```

- **Example 2:**

```
> y <- c(1.2, 3.4, 4.5, 6.4, 4, 3, 4, 3.4)
> f <- factor(c("A", "B", "B", "B", "A", "A", "B", "A"))
> n <- 8
> k <- 2
> Df <- c(k - 1, n - k)
> Df

[1] 1 6

> res <- levene.test(y, group = f)
> res$Df

[1] 1 6

> x <- abs(y - ave(y, f, FUN = "median"))
> Fvalue <- anova(lm(formula = x ~ f))$F
> Fvalue

[1] 0.01477833         NA

> res$"F value"

[1] 0.01477833         NA
```

```
> p.value <- 1 - pf(Fvalue, df1 = k - 1, df2 = n - k)
> p.value
```

```
[1] 0.9072118        NA
```

```
> res$"Pr(>F)"
```

```
[1] 0.9072118        NA
```

- **Example 3:**

```
> y <- c(4, 2.3, 8.8, 9.87, 12.34, 1.4, 6.4, 9.6, 8.86, 7.8, 8.6,
+     8.8, 2, 0.3)
> f <- factor(rep(c("Ctl", "Trt"), times = c(10, 4)))
> f
```

```
 [1] Ctl Ctl Ctl Ctl Ctl Ctl Ctl Ctl Ctl Ctl Trt Trt Trt Trt
Levels: Ctl Trt
```

```
> n <- 14
> k <- 2
> Df <- c(k - 1, n - k)
> Df
```

```
[1]  1 12
```

```
> res <- levene.test(y, group = f)
> res$Df
```

```
[1]  1 12
```

```
> x <- abs(y - ave(y, f, FUN = "median"))
> Fvalue <- anova(lm(formula = x ~ f))$F
> Fvalue
```

```
[1] 0.6701819        NA
```

```
> res$"F value"
```

```
[1] 0.6701819        NA
```

```
> p.value <- 1 - pf(Fvalue, df1 = k - 1, df2 = n - k)
> p.value
```

```
[1] 0.4289462        NA
```

```
> res$"Pr(>F)"
```

```
[1] 0.4289462        NA
```

10.5 Anova non parametrica a due fattori senza interazione

Test di Friedman

- **Package:** `stats`
- **Sintassi:** `friedman.test()`
- **Input:**

  `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`

$$df = k - 1$$

  `p.value`

$$P(\chi^2_{df} \geq c)$$

- **Example 1:**

```
> x <- matrix(c(6, 15, 8, 26, 29, 56, 60, 52, 20), nrow = 3, ncol = 3,
+     dimnames = list(NULL, c("X1", "X2", "X3")))
> x

     X1 X2 X3
[1,]  6 26 60
[2,] 15 29 52
[3,]  8 56 20

> n <- 3
> k <- 3
> matrice <- t(apply(x, MARGIN = 1, FUN = "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 <- colSums(matrice)[1]
> R2 <- colSums(matrice)[2]
> R3 <- colSums(matrice)[3]
> R <- c(R1, R2, R3)
> R

X1 X2 X3
 3  7  8
```

```
> statistic <- 12/(n * k * (k + 1)) * sum(R^2) - 3 * n * (k + 1)
> statistic

[1] 4.666667

> res <- friedman.test(x)
> res$statistic

Friedman chi-squared
            4.666667

> parameter <- k - 1
> parameter

[1] 2

> res$parameter

df
 2

> p.value <- 1 - pchisq(statistic, df = parameter)
> p.value

[1] 0.09697197

> res$p.value

[1] 0.09697197
```

- **Example 2:**

```
> x <- matrix(c(1, 3, 1, 3, 2, 2, 2, 3, 2, 3, 3, 1, 2, 1, 1), nrow = 5,
+     ncol = 3, dimnames = list(NULL, c("X1", "X2", "X3")))
> x

     X1 X2 X3
[1,]  1  2  3
[2,]  3  2  1
[3,]  1  3  2
[4,]  3  2  1
[5,]  2  3  1

> n <- 5
> k <- 3
> matrice <- t(apply(x, MARGIN = 1, FUN = "rank"))
> matrice

     X1 X2 X3
[1,]  1  2  3
[2,]  3  2  1
[3,]  1  3  2
[4,]  3  2  1
[5,]  2  3  1

> colSums(matrice)

X1 X2 X3
10 12  8
```

```
> R1 <- colSums(matrice)[1]
> R2 <- colSums(matrice)[2]
> R3 <- colSums(matrice)[3]
> R <- c(R1, R2, R3)
> R

X1 X2 X3
10 12  8

> statistic <- 12/(n * k * (k + 1)) * sum(R^2) - 3 * n * (k + 1)
> statistic

[1] 1.6

> res <- friedman.test(x)
> res$statistic

Friedman chi-squared
                 1.6

> parameter <- k - 1
> parameter

[1] 2

> res$parameter

df
 2

> p.value <- 1 - pchisq(statistic, df = parameter)
> p.value

[1] 0.449329

> res$p.value

[1] 0.449329
```

- **Example 3:**

```
> x <- matrix(0, nrow = 10, ncol = 6, byrow = TRUE, dimnames = list(NULL,
+     c("X1", "X2", "X3", "X4", "X5", "X6")))
> for (i in 1:10) x[i, ] <- sample(1:6)
> x

      X1 X2 X3 X4 X5 X6
 [1,]  5  3  4  2  6  1
 [2,]  3  1  4  2  6  5
 [3,]  1  4  5  3  2  6
 [4,]  3  1  6  2  5  4
 [5,]  6  2  5  4  3  1
 [6,]  6  4  5  2  3  1
 [7,]  1  4  2  3  5  6
 [8,]  1  6  3  2  5  4
 [9,]  6  2  1  5  4  3
[10,]  2  3  1  5  6  4
```

```
> n <- 10
> k <- 6
> matrice <- t(apply(x, MARGIN = 1, FUN = "rank"))
> matrice

      X1 X2 X3 X4 X5 X6
 [1,]  5  3  4  2  6  1
 [2,]  3  1  4  2  6  5
 [3,]  1  4  5  3  2  6
 [4,]  3  1  6  2  5  4
 [5,]  6  2  5  4  3  1
 [6,]  6  4  5  2  3  1
 [7,]  1  4  2  3  5  6
 [8,]  1  6  3  2  5  4
 [9,]  6  2  1  5  4  3
[10,]  2  3  1  5  6  4

> colSums(matrice)

X1 X2 X3 X4 X5 X6
34 30 36 30 45 35

> R1 <- colSums(matrice)[1]
> R2 <- colSums(matrice)[2]
> R3 <- colSums(matrice)[3]
> R4 <- colSums(matrice)[4]
> R5 <- colSums(matrice)[5]
> R6 <- colSums(matrice)[6]
> R <- c(R1, R2, R3, R4, R5, R6)
> R

X1 X2 X3 X4 X5 X6
34 30 36 30 45 35

> statistic <- 12/(n * k * (k + 1)) * sum(R^2) - 3 * n * (k + 1)
> statistic

[1] 4.342857

> res <- friedman.test(x)
> res$statistic

Friedman chi-squared
            4.342857

> parameter <- k - 1
> parameter

[1] 5

> res$parameter

df
 5

> p.value <- 1 - pchisq(statistic, df = parameter)
> p.value

[1] 0.5011797

> res$p.value

[1] 0.5011797
```

10.6 Test di ipotesi su una proporzione

Test di Bernoulli

- **Package:** `stats`
- **Sintassi:** `binom.test()`
- **Input:**

  `x` numero di successi

  `n` dimensione campionaria

  `p` valore di p_0

  `alternative = "less" / "greater" / "two.sided"` ipotesi alternativa

  `conf.level` livello di confidenza $1 - \alpha$

- **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
$x < n\,p_0$	$F_X(x) - F_X(n - y) + 1 \quad y = \#(p_X(k) \leq p_X(x) \quad \forall\, k = \lceil n\,p_0 \rceil, \ldots, n)$
$x > n\,p_0$	$F_X(y - 1) - F_X(x - 1) + 1 \quad y = \#(p_X(k) \leq p_X(x) \quad \forall\, k = 0, \ldots, \lfloor n\,p_0 \rfloor)$

$$X \sim Binomiale(n,\, p_0)$$

$$p_X(x) = \binom{n}{x} p_0^x \, (1 - p_0)^{n-x} \quad \forall\, x = 0, 1, \ldots, n$$

$$F_X(x) = \sum_{i=0}^{x} \binom{n}{i} p_0^i \, (1 - p_0)^{n-i} \quad \forall\, x = 0, 1, \ldots, n$$

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

$$p_0$$

- **Example 1:**

```
> x <- 682
> n <- 925
> p0 <- 0.75
> binom.test(x = 682, n = 925, p = 0.75, alternative = "two.sided",
+     conf.level = 0.95)$statistic

number of successes
                682

> binom.test(x = 682, n = 925, p = 0.75, alternative = "two.sided",
+     conf.level = 0.95)$parameter

number of trials
             925

> n * p0

[1] 693.75

> y <- sum(dbinom(ceiling(n * p0):n, n, p0) <= dbinom(x, n, p0))
> y

[1] 220

> p.value <- pbinom(x, n, p0) - pbinom(n - y, n, p0) + 1
> p.value

[1] 0.3824916

> binom.test(x = 682, n = 925, p = 0.75, alternative = "two.sided",
+     conf.level = 0.95)$p.value

[1] 0.3824916

> lower <- qbeta(0.025, x, n - x + 1)
> upper <- qbeta(0.975, x + 1, n - x)
> c(lower, upper)

[1] 0.7076683 0.7654066

> binom.test(x = 682, n = 925, p = 0.75, alternative = "two.sided",
+     conf.level = 0.95)$conf.int

[1] 0.7076683 0.7654066
attr(,"conf.level")
[1] 0.95
```

```
> x/n

[1] 0.7372973

> binom.test(x = 682, n = 925, p = 0.75, alternative = "two.sided",
+     conf.level = 0.95)$estimate

probability of success
             0.7372973

> p0

[1] 0.75

> binom.test(x = 682, n = 925, p = 0.75, alternative = "two.sided",
+     conf.level = 0.95)$null.value

probability of success
                  0.75
```

- **Example 2:**

```
> x <- 682
> n <- 925
> p0 <- 0.63
> binom.test(x = 682, n = 925, p = 0.63, alternative = "two.sided",
+     conf.level = 0.95)$statistic

number of successes
                682

> binom.test(x = 682, n = 925, p = 0.63, alternative = "two.sided",
+     conf.level = 0.95)$parameter

number of trials
             925

> n * p0

[1] 582.75

> y <- sum(dbinom(0:floor(n * p0), n, p0) <= dbinom(x, n, p0))
> y

[1] 480

> p.value <- pbinom(y - 1, n, p0) - pbinom(x - 1, n, p0) + 1
> p.value

[1] 4.925171e-12

> binom.test(x = 682, n = 925, p = 0.63, alternative = "two.sided",
+     conf.level = 0.95)$p.value

[1] 4.925209e-12

> ower <- qbeta(0.025, x, n - x + 1)
> upper <- qbeta(0.975, x + 1, n - x)
> c(lower, upper)
```

```
[1] 0.7076683 0.7654066

> binom.test(x = 682, n = 925, p = 0.63, alternative = "two.sided",
+     conf.level = 0.95)$conf.int

[1] 0.7076683 0.7654066
attr(,"conf.level")
[1] 0.95

> x/n

[1] 0.7372973

> binom.test(x = 682, n = 925, p = 0.63, alternative = "two.sided",
+     conf.level = 0.95)$estimate

probability of success
             0.7372973

> p0

[1] 0.63

> binom.test(x = 682, n = 925, p = 0.63, alternative = "two.sided",
+     conf.level = 0.95)$null.value

probability of success
                  0.63
```

10.7 Test di ipotesi sul ciclo di casualità

Test dei Runs

- **Package:** tseries
- **Sintassi:** runs.test()
- **Input:**

  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

$$z = \frac{V - \frac{n_1 + 2\,n_1\,n_2 + n_2}{n_1 + n_2}}{\sqrt{\frac{2\,n_1\,n_2\,(2\,n_1\,n_2 - n_1 - n_2)}{(n_1 + n_2)^2\,(n_1 + n_2 - 1)}}}$$

  p.value

- **Example 1:**

alternative	less	greater	two.sided
p.value	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\,|\,z\,|)$

```
> x <- factor(c("HIGH", "LOW", "LOW", "HIGH", "LOW", "HIGH", "HIGH",
+     "HIGH", "LOW", "HIGH", "HIGH", "LOW", "LOW", "HIGH", "LOW",
+     "HIGH", "LOW", "HIGH", "HIGH", "LOW", "HIGH", "LOW", "LOW",
+     "HIGH", "LOW", "HIGH", "HIGH", "LOW", "HIGH", "LOW"))
> x

 [1] HIGH LOW  LOW  HIGH LOW  HIGH HIGH HIGH LOW  HIGH HIGH LOW  LOW  HIGH LOW
[16] HIGH LOW  HIGH HIGH LOW  HIGH LOW  LOW  HIGH LOW  HIGH HIGH LOW  HIGH LOW
Levels: HIGH LOW

> n <- 30
> V <- 1 + sum(as.numeric(x[-1] != x[-n]))
> V

[1] 22

> n1 <- length(x[x == "HIGH"])
> n1

[1] 16

> n2 <- length(x[x == "LOW"])
> 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
```

- **Example 2:**

```
> x <- factor(c("a", "b", "b", "b", "a", "b", "b", "b", "a", "b",
+     "b", "b", "a", "a", "b", "b", "a", "a", "b", "b", "a", "b"))
> 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 <- 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
```

- **Example 3:**

```
> x <- factor(rep(1:2, each = 10))
> x

 [1] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
Levels: 1 2

> n <- 20
> V <- 1 + sum(as.numeric(x[-1] != x[-n]))
> V

[1] 2

> n1 <- length(x[x == "1"])
> n1

[1] 10

> n2 <- length(x[x == "2"])
> n2

[1] 10

> media <- (n1 + 2 * n1 * n2 + n2)/(n1 + n2)
> media

[1] 11

> varianza <- (2 * n1 * n2 * (2 * n1 * n2 - n1 - n2))/((n1 + n2)^2 *
+     (n1 + n2 - 1))
> varianza

[1] 4.736842

> z <- (V - media)/sqrt(varianza)
> z

[1] -4.135215

> runs.test(x, alternative = "two.sided")$statistic

Standard Normal
      -4.135215

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 3.546230e-05

> runs.test(x, alternative = "two.sided")$p.value

[1] 3.546230e-05
```

10.8 Test di ipotesi sulla differenza tra parametri di scala

Test di Mood

- **Package:** stats
- **Sintassi:** mood.test()
- **Input:**

  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

$$z = \frac{V - \frac{n_x\,(n_x+n_y+1)\,(n_x+n_y-1)}{12}}{\sqrt{\frac{n_x\,n_y\,(n_x+n_y+1)\,(n_x+n_y+2)\,(n_x+n_y-2)}{180}}}$$

  p.value

alternative	less	greater	two.sided
p.value	$\Phi(z)$	$1-\Phi(z)$	$2\,\Phi(-\mid z\mid)$

- **Example 1:**

```
> x <- c(-1, 1, -2, -1, 1, 1, 1, 1, -1, -2, 1, 1)
> y <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
> nx <- 12
> ny <- 9
> Rx <- rank(c(x, y))[1:nx]
> V <- 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 <- (V - 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
```

- **Example 2:**

```
> x <- c(1, 4.5, 6.78, 9.8, 7.7)
> y <- c(1, 4, 10, 2.1, 3.5, 5.6, 8.4, 12, 16.5, 22, 1.2, 3.4)
> nx <- 5
> ny <- 12
> Rx <- rank(c(x, y))[1:nx]
> V <- 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 <- (V - 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
```

- **Example 3:**

```
> x <- c(1, 1.2, 3.4, 0.8, 10.2, 9.3, 7.34)
> y <- c(-3.4, 0.2, 1.2, 2.1, 2.2, 2.2, 2.3, 3.1, 3.2, 4.2, 4.3,
+     5.43)
> nx <- 7
> ny <- 12
> Rx <- rank(c(x, y))[1:nx]
> V <- sum((Rx - (nx + ny + 1)/2)^2)
> media <- nx * (nx + ny + 1) * (nx + ny - 1)/12
> media

[1] 210

> varianza <- nx * ny * (nx + ny + 1) * (nx + ny + 2) * (nx + ny -
+     2)/180
> varianza

[1] 3332

> z <- (V - media)/sqrt(varianza)
> z

[1] 1.702080
```

```
> mood.test(x, y, alternative = "two.sided")$statistic

       Z
1.702080

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.0887403

> mood.test(x, y, alternative = "two.sided")$p.value

[1] 0.0887403
```

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: $\hat{n}_{ij} = n_{i\cdot}\, n_{\cdot j} \,/\, n_{\cdot\cdot} \quad \forall\, i = 1, 2, \ldots, h \quad \forall\, j = 1, 2, \ldots, k$
- frequenze attese nella m-esima tabella di contingenza 2×2:
  $\hat{n}_{ijm} = n_{i\cdot m}\, n_{\cdot jm} \,/\, n_{\cdot\cdot m} \quad \forall\, i, j = 1, 2 \quad \forall\, m = 1, 2, \ldots, l$
- 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 per tabelle di contingenza 2 righe per 2 colonne

Test Chi - Quadrato di indipendenza

- **Package:** `stats`
- **Sintassi:** `chisq.test()`
- **Input:**

  `x` matrice di dimensione 2×2 contenente frequenze assolute

  `correct = TRUE / FALSE` 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 = TRUE

$$c = \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{(|\, n_{ij} - \hat{n}_{ij} \,| - 1\,/\,2)^2}{\hat{n}_{ij}} = \frac{n_{..}\, (|\, n_{11}\, n_{22} - n_{12}\, n_{21} \,| - n_{..}\,/\,2)^2}{n_{1.}\, n_{2.}\, n_{.1}\, n_{.2}}$$

correct = FALSE

$$c = \sum_{i=1}^{2} \sum_{j=1}^{2} \frac{(n_{ij} - \hat{n}_{ij})^2}{\hat{n}_{ij}} = \frac{n_{..}\, (n_{11}\, n_{22} - n_{12}\, n_{21})^2}{n_{1.}\, n_{2.}\, n_{.1}\, n_{.2}}$$

parameter

$$df = 1$$

p.value

$$P(\chi^2_{df} \geq c)$$

observed

$$n_{ij} \quad \forall\, i, j = 1, 2$$

expected

$$\hat{n}_{ij} \quad \forall\, i, j = 1, 2$$

residuals

$$\frac{n_{ij} - \hat{n}_{ij}}{\sqrt{\hat{n}_{ij}}} \quad \forall\, i, j = 1, 2$$

- **Example 1:**

```
> x <- matrix(data = c(2, 10, 23, 21), nrow = 2, ncol = 2, byrow = FALSE)
> 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 = FALSE)

        Pearson's Chi-squared test

data:  x
X-squared = 4.8369, df = 1, p-value = 0.02786

> res <- chisq.test(x, correct = FALSE)
> res$statistic

X-squared
 4.836911

> res$parameter

df
 1

> res$p.value

[1] 0.02785675

> res$observed
```

```
   A  B
A  2 23
B 10 21
```

```
> res$expected
```

```
         A        B
A 5.357143 19.64286
B 6.642857 24.35714
```

```
> res$residuals
```

```
          A          B
A -1.450451  0.7574736
B  1.302544 -0.6802314
```

- **Example 2:**

```
> x <- matrix(data = c(2, 10, 23, 21), nrow = 2, ncol = 2, byrow = FALSE)
> 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 = TRUE)
```

```
        Pearson's Chi-squared test with Yates' continuity correction

data:  x
X-squared = 3.5034, df = 1, p-value = 0.06124
```

```
> res <- chisq.test(x, correct = TRUE)
> res$statistic
```

```
X-squared
 3.503421
```

```
> res$parameter
```

```
df
 1
```

```
> res$p.value
```

```
[1] 0.06124219
```

```
> res$observed
```

```
   A  B
A  2 23
B 10 21
```

```
> res$expected
```

```
         A        B
A 5.357143 19.64286
B 6.642857 24.35714

> res$residuals

          A          B
A -1.450451  0.7574736
B  1.302544 -0.6802314
```

- **Example 3:**

```
> x <- matrix(data = c(12, 5, 7, 7), nrow = 2, ncol = 2, byrow = FALSE)
> riga <- c("A", "B")
> colonna <- c("A", "B")
> dimnames(x) <- list(riga, colonna)
> x

   A B
A 12 7
B  5 7

> chisq.test(x, correct = TRUE)

        Pearson's Chi-squared test with Yates' continuity correction

data:  x
X-squared = 0.6411, df = 1, p-value = 0.4233

> res <- chisq.test(x, correct = TRUE)
> res$statistic

X-squared
0.6411203

> res$parameter

df
 1

> res$p.value

[1] 0.4233054

> res$observed

   A B
A 12 7
B  5 7

> res$expected

          A        B
A 10.419355 8.580645
B  6.580645 5.419355

> res$residuals

           A          B
A  0.4896818 -0.5396031
B -0.6161694  0.6789856
```

Test di McNemar

- **Package:** `stats`
- **Sintassi:** `mcnemar.test()`
- **Input:**

  `x` matrice di dimensione 2×2 contenente frequenze assolute

  `correct = TRUE / FALSE` correzione di *Yates*
- **Output:**

  `statistic` valore empirico della statistica χ^2

  `parameter` gradi di libertà

  `p.value` p-value
- **Formula:**

  `statistic`

  `correct = TRUE`

$$c = \frac{(|n_{12} - n_{21}| - 1)^2}{n_{12} + n_{21}}$$

  `correct = FALSE`

$$c = \frac{(n_{12} - n_{21})^2}{n_{12} + n_{21}}$$

  `parameter`

$$df = 1$$

  `p.value`

$$P(\chi^2_{df} \geq c)$$

- **Example 1:**

```
> x <- matrix(data = c(2, 10, 23, 21), nrow = 2, ncol = 2, byrow = FALSE)
> 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 = FALSE)

        McNemar's Chi-squared test

data:  x
McNemar's chi-squared = 5.1212, df = 1, p-value = 0.02364

> res <- mcnemar.test(x, correct = FALSE)
> res$statistic

McNemar's chi-squared
             5.121212

> res$parameter

df
 1
```

```
> res$p.value

[1] 0.0236351
```

- **Example 2:**

```
> x <- matrix(data = c(2, 10, 23, 21), nrow = 2, ncol = 2, byrow = FALSE)
> 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 = TRUE)

        McNemar's Chi-squared test with continuity correction

data:  x
McNemar's chi-squared = 4.3636, df = 1, p-value = 0.03671

> res <- mcnemar.test(x, correct = TRUE)
> res$statistic

McNemar's chi-squared
             4.363636

> res$parameter

df
 1

> res$p.value

[1] 0.03671386
```

- **Example 3:**

```
> x <- matrix(data = c(12, 5, 7, 7), nrow = 2, ncol = 2, byrow = FALSE)
> riga <- c("A", "B")
> colonna <- c("A", "B")
> dimnames(x) <- list(riga, colonna)
> x

   A B
A 12 7
B  5 7

> mcnemar.test(x, correct = TRUE)

        McNemar's Chi-squared test with continuity correction

data:  x
McNemar's chi-squared = 0.0833, df = 1, p-value = 0.7728

> res <- mcnemar.test(x, correct = TRUE)
> res$statistic
```

```
McNemar's chi-squared
           0.08333333

> res$parameter

df
 1

> res$p.value

[1] 0.77283
```

Test esatto di Fisher

- **Package:** `stats`
- **Sintassi:** `fisher.test()`
- **Input:**

  `x` matrice di dimensione 2×2 contenente frequenze assolute

  `alternative = "less" / "greater" / "two.sided"` ipotesi alternativa

- **Output:**

  `p.value` p-value

  `alternative` ipotesi alternativa

- **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	$\sum_{i=0}^{n_{11}} p(i) + \sum_{p(i) \leq p(n_{11})} p(i) \quad \forall\, i = n_{11}+1, \ldots, \min(n_{1\cdot}, n_{\cdot 1})$

$$p(i) = \frac{{}_{\max(n_{1\cdot},\, n_{\cdot 1})}C_i \quad {}_{n_{\cdot\cdot} - \max(n_{1\cdot},\, n_{\cdot 1})}C_{\min(n_{1\cdot},\, n_{\cdot 1}) - i}}{{}_{n}C_{\min(n_{1\cdot},\, n_{\cdot 1})}} \quad \forall\, i = 0, 1, \ldots, \min(n_{1\cdot}, n_{\cdot 1})$$

- **Example 1:**

```
> x <- matrix(data = c(2, 9, 5, 4), nrow = 2, ncol = 2, byrow = FALSE)
> 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
```

- **Example 2:**

```
> x <- matrix(data = c(3, 7, 6, 5), nrow = 2, ncol = 2, byrow = FALSE)
> riga <- c("A", "B")
> colonna <- c("A", "B")
> dimnames(x) <- list(riga, colonna)
> x

  A B
A 3 6
B 7 5

> n11 <- 3
> n1. <- 3 + 6
> n.1 <- 3 + 7
> n.. <- 3 + 6 + 7 + 5
> n..

[1] 21

> minimo <- min(n1., n.1)
> minimo

[1] 9

> massimo <- max(n1., n.1)
> massimo

[1] 10

> 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] 0.2449393

> fisher.test(x, alternative = "less")$p.value

[1] 0.2449393

> 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] 0.943677

> fisher.test(x, alternative = "greater")$p.value

[1] 0.943677

> p.value1 <- 0
> for (i in 0:n11) p.value1 <- p.value1 + p(i)
> p.value1

[1] 0.2449393
```

```
> p.value2 <- 0
> for (i in (n11 + 1):minimo) {
+     if (p(i) <= p(n11))
+         p.value2 <- p.value2 + p(i)
+ }
> p.value2

[1] 0.1420576

> p.value.two.sided <- p.value1 + p.value2
> p.value.two.sided

[1] 0.3869969

> fisher.test(x, alternative = "two.sided")$p.value

[1] 0.3869969
```

- **Example 3:**

```
> x <- matrix(c(2, 9, 3, 4), nrow = 2, ncol = 2, byrow = FALSE)
> riga <- c("A", "B")
> colonna <- c("A", "B")
> dimnames(x) <- list(riga, colonna)
> x

  A B
A 2 3
B 9 4

> n11 <- 2
> n1. <- 2 + 3
> n.1 <- 2 + 9
> n.. <- 2 + 3 + 9 + 4
> n..

[1] 18

> minimo <- min(n1., n.1)
> minimo

[1] 5

> 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] 0.2720588

> fisher.test(x, alternative = "less")$p.value

[1] 0.2720588
```

```
> 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] 0.9526144

> fisher.test(x, alternative = "greater")$p.value

[1] 0.9526144

> p.value1 <- 0
> for (i in 0:n11) p.value1 <- p.value1 + p(i)
> p.value1

[1] 0.2720588

> p.value2 <- 0
> for (i in (n11 + 1):minimo) {
+     if (p(i) <= p(n11))
+         p.value2 <- p.value2 + p(i)
+ }
> p.value2

[1] 0.05392157

> p.value.two.sided <- p.value1 + p.value2
> p.value.two.sided

[1] 0.3259804

> fisher.test(x, alternative = "two.sided")$p.value

[1] 0.3259804
```

Test di Mantel - Haenszel

- **Package:** `stats`
- **Sintassi:** `mantelhaen.test()`
- **Input:**

  `x` array di dimensione $2 \times 2 \times l$ contenente l tabelle di contingenza 2×2

  `conf.level` livello di confidenza $1 - \alpha$

  `correct = FALSE`

- **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`

$$c = \frac{\left[\sum_{m=1}^{l} \left(n_{11m} - \hat{n}_{11m}\right)\right]^2}{\sum_{m=1}^{l} \hat{\sigma}^2_{n_{11m}}}$$

$$\text{dove} \quad \hat{\sigma}^2_{n_{11m}} = \frac{n_{1\cdot m}\, n_{2\cdot m}\, n_{\cdot 1m}\, n_{\cdot 2m}}{n^2_{\cdot\cdot m}\,(n_{\cdot\cdot m} - 1)} \quad \forall\, m = 1, 2, \ldots, l$$

`parameter`

$$df = 1$$

`p.value`

$$P(\chi^2_{df} \geq c)$$

`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

$$\hat{\sigma}^2_{\log(\hat{\theta}_{MH})} = \frac{1}{R^2} \sum_{m=1}^{l} \frac{(n_{11m} + n_{22m})\, R_m}{n_{\cdot\cdot m}} + \frac{1}{S^2} \sum_{m=1}^{l} \frac{(n_{12m} + n_{21m})\, S_m}{n_{\cdot\cdot m}} + \\ + \frac{1}{2\,R\,S} \sum_{m=1}^{l} \frac{(n_{11m} + n_{22m})\, S_m + (n_{12m} + n_{21m})\, R_m}{n_{\cdot\cdot m}}$$

- **Examples:**

```
> 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(x, conf.level = 0.95, correct = FALSE)

        Mantel-Haenszel chi-squared test without continuity correction

data:  x
Mantel-Haenszel X-squared = 6.3841, df = 1, p-value = 0.01151
alternative hypothesis: true common odds ratio is not equal to 1
95 percent confidence interval:
 1.177590 3.869174
sample estimates:
common odds ratio
         2.134549

> res <- mantelhaen.test(x, conf.level = 0.95, correct = FALSE)
> res$statistic

Mantel-Haenszel X-squared
                 6.384113

> res$parameter

df
 1

> res$p.value

[1] 0.01151463

> res$estimate

common odds ratio
         2.134549
```

```
> res$conf.int

[1] 1.177590 3.869174
attr(,"conf.level")
[1] 0.95
```

11.3 Test di ipotesi per tabelle di contingenza n righe per k colonne

Test Chi - Quadrato di indipendenza

- **Package:** stats
- **Sintassi:** chisq.test()
- **Input:**

  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

$$c = \sum_{i=1}^{h} \sum_{j=1}^{k} \frac{(n_{ij} - \hat{n}_{ij})^2}{\hat{n}_{ij}} = \sum_{i=1}^{h} \sum_{j=1}^{k} \frac{n_{ij}^2}{\hat{n}_{ij}} - n_{\cdot\cdot} = n_{\cdot\cdot} \left(\sum_{i=1}^{h} \sum_{j=1}^{k} \frac{n_{ij}^2}{n_{i\cdot}\, n_{\cdot j}} - 1 \right)$$

  parameter

$$df = (h-1)\,(k-1)$$

  p.value

$$P(\chi^2_{df} \geq c)$$

  observed

$$n_{ij} \quad \forall\, i = 1, 2, \dots, h \quad \forall\, j = 1, 2, \dots, k$$

  expected

$$\hat{n}_{ij} \quad \forall\, i = 1, 2, \dots, h \quad \forall\, j = 1, 2, \dots, k$$

  residuals

$$\frac{n_{ij} - \hat{n}_{ij}}{\sqrt{\hat{n}_{ij}}} \quad \forall\, i = 1, 2, \dots, h \quad \forall\, j = 1, 2, \dots, k$$

- **Examples:**

```
> x <- matrix(data = c(2, 10, 23, 21, 11, 12, 43, 32, 30), nrow = 3,
+     ncol = 3)
> riga <- c("A", "B", "C")
> colonna <- c("A", "B", "C")
> dimnames(x) <- list(riga, colonna)
> x

   A  B  C
A  2 21 43
B 10 11 32
C 23 12 30
```

```
> h <- 3
> k <- 3
> chisq.test(x)

        Pearson's Chi-squared test

data:  x
X-squared = 22.9907, df = 4, p-value = 0.0001272

> res <- chisq.test(x)
> res$statistic

X-squared
 22.99074

> res$parameter

df
 4

> res$p.value

[1] 0.0001271668

> res$observed

   A  B  C
A  2 21 43
B 10 11 32
C 23 12 30

> res$expected

         A        B        C
A 12.55435 15.78261 37.66304
B 10.08152 12.67391 30.24457
C 12.36413 15.54348 37.09239

> res$residuals

            A          B          C
A -2.97875184  1.3133002  0.8696329
B -0.02567500 -0.4701945  0.3191986
C  3.02476204 -0.8987847 -1.1645289
```

Test di McNemar

- **Package:** `stats`
- **Sintassi:** `mcnemar.test()`
- **Input:**

  `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`

$$P(\chi^2_{df} \geq c)$$

- **Examples:**

```
> x <- matrix(data = c(2, 10, 23, 21, 11, 12, 43, 32, 30), nrow = 3,
+     ncol = 3)
> riga <- c("A", "B", "C")
> colonna <- c("A", "B", "C")
> dimnames(x) <- list(riga, colonna)
> x


   A  B  C
A  2 21 43
B 10 11 32
C 23 12 30


> n <- 3
> mcnemar.test(x)


        McNemar's Chi-squared test

data:  x
McNemar's chi-squared = 19.0547, df = 3, p-value = 0.0002664


> res <- mcnemar.test(x)
> res$statistic


McNemar's chi-squared
             19.05474


> res$parameter


df
 3


> res$p.value


[1] 0.0002663652
```

11.4 Comandi utili per le tabelle di contingenza

margin.table()

- **Package:** base
- **Input:**

  x matrice di dimensione $h \times k$ contenente frequenze assolute

  margin = NULL / 1 / 2 marginale assoluto totale, di riga o di colonna

- **Description:** distribuzione marginale assoluta
- **Formula:**

margin = NULL

$$n_{\cdot\cdot}$$

margin = 1

$$n_{i\cdot} \quad \forall\, i = 1, 2, \ldots, h$$

margin = 2

$$n_{\cdot j} \quad \forall\, j = 1, 2, \ldots, k$$

- **Example 1:**

```
> x <- matrix(data = c(1, 3, 0, 1, 3, 2, 2, 1, 2), nrow = 3, ncol = 3,
+     byrow = TRUE)
> riga <- c("a", "b", "c")
> colonna <- c("A", "B", "C")
> dimnames(x) <- list(riga, colonna)
> x

  A B C
a 1 3 0
b 1 3 2
c 2 1 2

> h <- 3
> k <- 3
> margin.table(x, margin = NULL)

[1] 15
```

- **Example 2:**

```
> x <- matrix(data = c(1, 3, 0, 1, 3, 2, 2, 1, 2), nrow = 3, ncol = 3,
+     byrow = TRUE)
> riga <- c("a", "b", "c")
> colonna <- c("A", "B", "C")
> dimnames(x) <- list(riga, colonna)
> x

  A B C
a 1 3 0
b 1 3 2
c 2 1 2

> h <- 3
> k <- 3
```

- **Example 3:**

```
> x <- matrix(data = c(1, 3, 0, 1, 3, 2, 2, 1, 2), nrow = 3, ncol = 3,
+     byrow = TRUE)
> riga <- c("a", "b", "c")
> colonna <- c("A", "B", "C")
> dimnames(x) <- list(riga, colonna)
> x

  A B C
a 1 3 0
b 1 3 2
c 2 1 2

> h <- 3
> k <- 3
> margin.table(x, margin = 1)

a b c
4 6 5

> margin.table(x, margin = 2)

A B C
4 7 4
```

prop.table()

- **Package:** base
- **Input:**

  x matrice di dimensione $h \times k$ contenente frequenze assolute

  margin = NULL / 1 / 2 frequenza relativa totale, di riga o di colonna

- **Description:** distribuzione relativa
- **Formula:**

margin = NULL

$$n_{ij} \,/\, n_{\cdot\cdot} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, h \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k$$

margin = 1

$$n_{ij} \,/\, n_{i\cdot} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, h \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k$$

margin = 2

$$n_{ij} \,/\, n_{\cdot j} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, h \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k$$

- **Example 1:**

```
> x <- matrix(data = c(1, 3, 0, 1, 3, 2, 2, 1, 2), nrow = 3, ncol = 3,
+     byrow = TRUE)
> riga <- c("a", "b", "c")
> colonna <- c("A", "B", "C")
> dimnames(x) <- list(riga, colonna)
> x
```

```
  A B C
a 1 3 0
b 1 3 2
c 2 1 2
```

```
> h <- 3
> k <- 3
> 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
```

- **Example 2:**

```
> x <- matrix(data = c(1, 3, 0, 1, 3, 2, 2, 1, 2), nrow = 3, ncol = 3,
+     byrow = TRUE)
> riga <- c("a", "b", "c")
> colonna <- c("A", "B", "C")
> dimnames(x) <- list(riga, colonna)
> x
```

```
  A B C
a 1 3 0
b 1 3 2
c 2 1 2
```

```
> h <- 3
> k <- 3
> 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
```

- **Example 3:**

```
> x <- matrix(data = c(1, 3, 0, 1, 3, 2, 2, 1, 2), nrow = 3, ncol = 3,
+     byrow = TRUE)
> riga <- c("a", "b", "c")
> colonna <- c("A", "B", "C")
> dimnames(x) <- list(riga, colonna)
> x
```

```
  A B C
a 1 3 0
b 1 3 2
c 2 1 2
```

```
> h <- 3
> k <- 3
> 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()

- **Package:** stats
- **Input:**

  y vettore numerico di dimensione n

  f fattore a k livelli

  g fattore a h livelli

- **Description:** costruzione di una tabella di contingenza a partire da un dataframe
- **Examples:**

```
> y <- c(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 <- factor(rep(letters[1:2], each = 6))
> f

 [1] a a a a a a b b b b b b
Levels: a b

> g <- factor(rep(LETTERS[2:1], times = 6))
> g

 [1] B A B A B A B A B A B A
Levels: A B

> data.frame(f, g, y)

   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()

- **Package:** stats
- **Input:**

  x oggetto di tipo table contenente frequenze assolute

  row.vars variabili di riga

  col.vars variabili di colonna

- **Description:** costruzione di flat tables

- **Examples:**

```
> 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()

- **Package:** `base`
- **Input:**

  `x` oggetto di tipo `table` di dimensione $h \times k$ contenente frequenze assolute
- **Description:** 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`

$$c = \sum_{i=1}^{h} \sum_{j=1}^{k} \frac{(n_{ij} - \hat{n}_{ij})^2}{\hat{n}_{ij}} = n_{..} \left(\sum_{i=1}^{h} \sum_{j=1}^{k} \frac{n_{ij}^2}{n_{i\cdot}\, n_{\cdot j}} - 1 \right)$$

  `parameter`

$$df = (h-1)\,(k-1)$$

  `p.value`

$$P(\chi^2_{df} \geq c)$$

- **Example 1:**

```
> f <- factor(c("a", "b", "c", "b", "a", "c", "a", "b", "b", "c",
+     "a"))
> f

 [1] a b c b a c a b b c a
Levels: a b c

> g <- factor(c("A", "S", "A", "S", "S", "S", "A", "S", "S", "A",
+     "A"))
> 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 <- 3
> k <- 2
> summary(x)

Number of cases in table: 11
Number of factors: 2
Test for independence of all factors:
        Chisq = 5.286, df = 2, p-value = 0.07114
        Chi-squared approximation may be incorrect

> res <- summary(x)
> res$n.cases

[1] 11

> res$statistic

[1] 5.286111

> res$parameter

[1] 2

> res$p.value

[1] 0.07114355
```

- **Example 2:**

```
> f <- factor(c("a", "b", "a", "b", "a", "a", "b", "b", "a", "b",
+     "a"))
> f

 [1] a b a b a a b b a b a
Levels: a b

> g <- factor(c("A", "S", "A", "S", "S", "S", "A", "S", "S", "A",
+     "A"))
> 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 3
  b 2 3

> h <- 2
> k <- 2
> summary(x)

Number of cases in table: 11
Number of factors: 2
Test for independence of all factors:
        Chisq = 0.11, df = 1, p-value = 0.7401
        Chi-squared approximation may be incorrect
```

```
> res <- summary(x)
> res$n.cases

[1] 11

> res$statistic

[1] 0.11

> res$parameter

[1] 1

> res$p.value

[1] 0.7401441
```

Capitolo 12

Test di ipotesi sull'adattamento

12.1 Test di ipotesi sulla distribuzione normale

Test di Kolmogorov - Smirnov

- **Package:** stats
- **Sintassi:** `ks.test()`
- **Input:**

  `x` vettore numerico di n valori distinti

- **Description:** 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`

$$d = \max_{1 \leq i \leq n} \left\{ \max \left[\frac{i}{n} - F_0\left(x_{(i)}\right),\, F_0\left(x_{(i)}\right) - \frac{i-1}{n} \right] \right\}$$

$$\text{dove} \quad F_0\left(x_{(i)}\right) = \Phi\left(\frac{x_{(i)} - \mu}{\sigma}\right) \quad \forall\, i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(0.1, 2.3, 4.3, 4.2, 5.6, 7.21, 8.2)
> n <- 7
> x <- sort(x)
> x

[1] 0.10 2.30 4.20 4.30 5.60 7.21 8.20

> Fo <- pnorm(x, mean = 3.3, sd = 1.2)
> 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
```

- **Example 2:**

```
> x <- c(1.1, 3.4, 5.6, 7.8, 2.3, 4.5, 1.2, 2.2)
> n <- 8
> x <- sort(x)
> x

[1] 1.1 1.2 2.2 2.3 3.4 4.5 5.6 7.8

> Fo <- pnorm(x, mean = 4.1, sd = 2.3)
> 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
```

- **Example 3:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.8)
> n <- 8
> x <- sort(x)
> x

[1] 1.1 2.3 3.4 4.5 5.6 6.7 6.8 8.9

> Fo <- pnorm(x, mean = 6.3, sd = 1.1)
> vettore1 <- (1:n)/n - Fo
> vettore2 <- Fo - ((1:n) - 1)/n
> d <- max(pmax(vettore1, vettore2))
> d

[1] 0.4491182

> ks.test(x, "pnorm", 6.3, 1.1)$statistic

        D
0.4491182
```

Test di Jarque - Bera

- **Package:** `tseries`
- **Sintassi:** `jarque.bera.test()`
- **Input:**

  `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

$$df = 2$$

p.value

$$P(\chi^2_{df} \geq c)$$

- **Example 1:**

```
> x <- c(0.1, 2.3, 4.3, 4.2, 5.6, 7.21, 8.2)
> n <- 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
```

- **Example 2:**

```
> x <- c(1.1, 3.4, 5.6, 7.8, 2.3, 4.5, 1.2, 2.2, 1.1)
> n <- 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
```

- **Example 3:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> n <- 8
> m2 <- mean((x - mean(x))^2)
> m2

[1] 5.8225

> m3 <- mean((x - mean(x))^3)
> m3

[1] 0.015

> m4 <- mean((x - mean(x))^4)
> m4
```

```
[1] 67.06683

> c <- (n/6) * (m3/m2^(3/2))^2 + (n/24) * (m4/m2^2 - 3)^2
> c

[1] 0.347969

> jarque.bera.test(x)$statistic

X-squared
 0.347969

> jarque.bera.test(x)$parameter

df
 2

> p.value <- 1 - pchisq(c, df = 2)
> p.value

[1] 0.8403099

> jarque.bera.test(x)$p.value

X-squared
0.8403099
```

Test di Cramer - von Mises

- **Package:** nortest
- **Sintassi:** cvm.test()
- **Input:**

  x vettore numerico di dimensione $n \geq 7$

- **Output:**

  statistic valore empirico della statistica Z

  p.value p-value

- **Formula:**

  statistic

$$W = \frac{1}{12\,n} + \sum_{i=1}^{n} \left[\Phi\left(\frac{x_{(i)} - \bar{x}}{s_x} \right) - \frac{2\,i - 1}{2\,n} \right]^2$$

  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	$e^{0.886 - 31.62\,WW + 10.897\,WW^2}$	$e^{1.111 - 34.242\,WW + 12.832\,WW^2}$

- **Example 1:**

```
> x <- c(1.1, 1.2, 2.2, 2.3, 3.4, 4.5, 5.6, 7.8)
> n <- 8
> x <- sort(x)
> 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

> 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
```

- **Example 2:**

```
> x <- c(80, 96.19, 98.07, 99.7, 99.79, 99.81, 101.14, 101.6, 103.44,
+     103.53)
> n <- 10
> x <- sort(x)
> 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

> 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
```

- **Example 3:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> n <- 8
> x <- sort(x)
> W <- (1/(12 * n)) + sum((pnorm((x - mean(x))/sd(x)) - (2 * (1:n) -
+     1)/(2 * n))^2)
> W

[1] 0.02235135

> cvm.test(x)$statistic

         W
0.02235135

> WW <- (1 + 0.5/n) * W
> WW

[1] 0.02374831

> p.value <- 1 - exp(-13.953 + 775.5 * WW - 12542.61 * WW^2)
> p.value

[1] 0.9264651

> cvm.test(x)$p.value

[1] 0.9264651
```

Test di Anderson - Darlin

- **Package:** nortest
- **Sintassi:** `ad.test()`
- **Input:**

  `x` vettore numerico di dimensione $n \geq 7$
- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value
- **Formula:**

  `statistic`

$$A = -n - \frac{1}{n} \sum_{i=1}^{n} (2\,i - 1) \left[\log\left(\Phi\left(\frac{x_{(i)} - \bar{x}}{s_x} \right) \right) + \log\left(1 - \Phi\left(\frac{x_{(n-i+1)} - \bar{x}}{s_x} \right) \right) \right]$$

  `p.value`

$$AA = (1 + 0.75 \,/\, n + 2.25 \,/\, n^2)\, A$$

- **Example 1:**

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	$e^{0.9177-4.279\,AA-1.38\,AA^2}$	$e^{1.2937-5.709\,AA+0.0186\,AA^2}$

```
> x <- c(99.7, 99.79, 101.14, 99.32, 99.27, 101.29, 100.3, 102.4,
+     105.2)
> n <- 9
> x <- sort(x)
> 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

> 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
```

- **Example 2:**

```
> x <- c(1.1, 1.2, 2.2, 2.3, 3.4, 4.5, 5.6, 7.8)
> n <- 8
> x <- sort(x)
> 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

> 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
```

- **Example 3:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> n <- 8
> x <- sort(x)
> 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] 0.1546968

> ad.test(x)$statistic

        A
0.1546968

> AA <- (1 + 0.75/n + 2.25/n^2) * A
> AA

[1] 0.1746381

> p.value <- 1 - exp(-13.436 + 101.14 * AA - 223.73 * AA^2)
> p.value

[1] 0.9254678

> ad.test(x)$p.value

[1] 0.9254678
```

Test di Shapiro - Francia

- **Package:** nortest
- **Sintassi:** sf.test()
- **Input:**

  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`

$$1 - \Phi(z)$$

$$\text{dove} \quad z = \frac{\log(1 - W) - [-1.2725 + 1.0521\,[\log(\log(n)) - \log(n)]]}{1.0308 - 0.26758\,[\log(\log(n)) + 2\,/\,\log(n)]}$$

- **Example 1:**

```
> x <- c(7.7, 5.6, 4.3, 3.2, 3.1, 2.2, 1.2, 1)
> n <- 8
> x <- sort(x)
> 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
```

- **Example 2:**

```
> x <- c(1.2, 3.2, 4.2, 2.1, 0.34, 3.4, 9.3, 9.2, 9.9, 10.2, 11.2)
> n <- 11
> x <- sort(x)
> 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
```

- **Example 3:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> n <- 8
> x <- sort(x)
> y <- qnorm(((1:n) - 3/8)/(n + 1/4))
> W <- cor(x, y)^2
> W

[1] 0.9838034

> sf.test(x)$statistic

        W
0.9838034

> 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] -2.48103

> p.value <- 1 - pnorm(z)
> p.value

[1] 0.9934498

> sf.test(x)$p.value

[1] 0.9934498
```

Test di Lilliefors

- **Package:** nortest
- **Sintassi:** `lillie.test()`
- **Input:**

  `x` vettore numerico di dimensione $n \geq 5$
- **Output:**

  `statistic` valore empirico della statistica Z

  `p.value` p-value
- **Formula:**

n	$n \leq 100$	$n > 100$
Kd	D	$(n \,/\, 100)^{0.49}\, D$
nd	n	100

`statistic`

$$D \;=\; \max(a,\, b)$$

dove
$$\begin{aligned} \text{a} &= \max\left\{\frac{i}{n} - \Phi\left(\frac{x_{(i)} - \bar{x}}{s_x}\right)\right\}_{i\,=\,1,\,2,\,\dots,\,n} \\ \text{b} &= \max\left\{\Phi\left(\frac{x_{(i)} - \bar{x}}{s_x}\right) - \frac{i-1}{n}\right\}_{i\,=\,1,\,2,\,\dots,\,n} \end{aligned}$$

`p.value`

$$pvalue \;=\; e^{-7.01256\,Kd^2\,(nd+2.78019)\,+2.99587\,Kd\,\sqrt{nd+2.78019}-0.122119+\frac{0.974598}{\sqrt{nd}}+\frac{1.67997}{nd}}$$

$$\boxed{pvalue \leq 0.1}$$

$$\texttt{p.value} = pvalue$$

$$\boxed{pvalue > 0.1}$$

$$kk \;=\; (\sqrt{n} - 0.01 + 0.85 \,/\, \sqrt{n})\; D$$

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

- **Example 1:**

```
> x <- c(1.1, 1.2, 2.2, 2.3, 3.4, 4.5, 5.6, 7.8)
> n <- 8
> x <- sort(x)
> 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
```

```
> 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

> kk <- (sqrt(n) - 0.01 + 0.85/sqrt(n)) * D
> kk

[1] 0.6187895

> 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
```

- **Example 2:**

```
> x <- c(42.3, 31.4, 11.2, 9, 8.5, 7.5, 5.6, 2.3)
> n <- 8
> x <- sort(x)
> 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

> 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

> p.value <- pvalue
> p.value

[1] 0.004993897
```

```
> lillie.test(x)$p.value

[1] 0.004993897
```

- **Example 3:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> n <- 8
> x <- sort(x)
> a <- max((1:n)/n - pnorm((x - mean(x))/sd(x)))
> a

[1] 0.1176558

> b <- max(pnorm((x - mean(x))/sd(x)) - ((1:n) - 1)/n)
> b

[1] 0.1323442

> D <- max(a, b)
> D

[1] 0.1323442

> lillie.test(x)$statistic

        D
0.1323442

> 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] 1.507065

> kk <- (sqrt(n) - 0.01 + 0.85/sqrt(n)) * D
> kk

[1] 0.4127748

> p.value <- 2.76773 - 19.828315 * kk + 80.709644 * kk^2 - 138.55152 *
+     kk^3 + 81.218052 * kk^4
> p.value

[1] 0.9481423

> lillie.test(x)$p.value

[1] 0.9481423
```

Test di Anscombe - Glynn

- **Package:** moments
- **Sintassi:** `anscombe.test()`
- **Input:**

  `x` vettore numerico 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`

$$z = \frac{1 - \frac{2}{9\,a} - \left(\frac{1 - 2\,/\,a}{1 + xx\,\sqrt{2\,/\,(a-4)}} \right)^{1\,/\,3}}{\sqrt{\frac{2}{9\,a}}}$$

dove

$$\begin{aligned} b &= \frac{1}{n} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{\sigma_x} \right)^4 \\ eb2 &= \frac{3\,(n-1)}{(n+1)} \\ vb2 &= \frac{24\,n\,(n-2)\,(n-3)}{(n+1)^2\,(n+3)\,(n+5)} \\ m3 &= \frac{6\,(n^2 - 5\,n + 2)}{(n+7)\,(n+9)} \sqrt{\frac{6\,(n+3)\,(n+5)}{n\,(n-2)\,(n-3)}} \\ a &= 6 + \frac{8}{m3} \left(\frac{2}{m3} + \sqrt{1 + \frac{4}{m3}} \right) \\ xx &= (b - eb2)\,/\,\sqrt{vb2} \end{aligned}$$

  `p.value`

alternative	less	greater	two.sided
p.value	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\,\vert\,z\,\vert)$

- **Example 1:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1, 6.7, 7.6, 6.8)
> n <- length(x)
> b <- n * sum((x - mean(x))^4)/(sum((x - mean(x))^2)^2)
> eb2 <- 3 * (n - 1)/(n + 1)
> vb2 <- 24 * n * (n - 2) * (n - 3)/((n + 1)^2 * (n + 3) * (n +
+     5))
> m3 <- (6 * (n^2 - 5 * n + 2)/((n + 7) * (n + 9))) * sqrt((6 *
+     (n + 3) * (n + 5))/(n * (n - 2) * (n - 3)))
> a <- 6 + (8/m3) * (2/m3 + sqrt(1 + 4/m3))
> xx <- (b - eb2)/sqrt(vb2)
> res <- anscombe.test(x, alternative = "two.sided")
> z <- (1 - 2/(9 * a) - ((1 - 2/a)/(1 + xx * sqrt(2/(a - 4))))^(1/3))/sqrt(2/(9 *
+     a))
> c(b, z)

[1]  1.8382073 -0.9304068
```

```
> res$statistic

     kurt          z
 1.8382073 -0.9304068

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.3521605

> res$p.value

[1] 0.3521605
```

- **Example 2:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> n <- length(x)
> b <- n * sum((x - mean(x))^4)/(sum((x - mean(x))^2)^2)
> eb2 <- 3 * (n - 1)/(n + 1)
> vb2 <- 24 * n * (n - 2) * (n - 3)/((n + 1)^2 * (n + 3) * (n +
+     5))
> m3 <- (6 * (n^2 - 5 * n + 2)/((n + 7) * (n + 9))) * sqrt((6 *
+     (n + 3) * (n + 5))/(n * (n - 2) * (n - 3)))
> a <- 6 + (8/m3) * (2/m3 + sqrt(1 + 4/m3))
> xx <- (b - eb2)/sqrt(vb2)
> res <- anscombe.test(x, alternative = "two.sided")
> z <- (1 - 2/(9 * a) - ((1 - 2/a)/(1 + xx * sqrt(2/(a - 4))))^(1/3))/sqrt(2/(9 *
+     a))
> c(b, z)

[1]  1.623612 -0.734540

> res$statistic

    kurt         z
 1.623612 -0.734540

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.4626197

> res$p.value

[1] 0.4626197
```

- **Example 3:**

```
> x <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- length(x)
> b <- n * sum((x - mean(x))^4)/(sum((x - mean(x))^2)^2)
> eb2 <- 3 * (n - 1)/(n + 1)
> vb2 <- 24 * n * (n - 2) * (n - 3)/((n + 1)^2 * (n + 3) * (n +
+     5))
> m3 <- (6 * (n^2 - 5 * n + 2)/((n + 7) * (n + 9))) * sqrt((6 *
+     (n + 3) * (n + 5))/(n * (n - 2) * (n - 3)))
> a <- 6 + (8/m3) * (2/m3 + sqrt(1 + 4/m3))
> xx <- (b - eb2)/sqrt(vb2)
> res <- anscombe.test(x, alternative = "two.sided")
> z <- (1 - 2/(9 * a) - ((1 - 2/a)/(1 + xx * sqrt(2/(a - 4))))^(1/3))/sqrt(2/(9 *
+     a))
> c(b, z)
```

```
[1] 4.726207 2.449794

> res$statistic

    kurt        z
4.726207 2.449794

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.01429380

> res$p.value

[1] 0.01429380
```

Test di Bonett - Seier

- **Package:** moments
- **Sintassi:** bonett.test()
- **Input:**

  x vettore numerico 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

$$z = \sqrt{n+2}\,(13.29 \log(\rho \,/\, \tau) - 3) \,/\, 3.54$$

$$\text{dove} \quad \rho = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad \text{e} \quad \tau = \frac{1}{n}\sum_{i=1}^{n}|\,x_i - \bar{x}\,|$$

  p.value

alternative	less	greater	two.sided
p.value	$\Phi(z)$	$1 - \Phi(z)$	$2\,\Phi(-\,|\,z\,|)$

- **Example 1:**

```
> x <- c(7.8, 6.6, 6.5, 7.4, 7.3, 7, 6.4, 7.1, 6.7, 7.6, 6.8)
> n <- length(x)
> rho <- sqrt((n - 1) * var(x)/n)
> tau <- mean(abs(x - mean(x)))
> res <- bonett.test(x, alternative = "two.sided")
> z <- sqrt(n + 2) * (13.29 * log(rho/tau) - 3)/3.54
> c(tau, z)

[1]  0.3834711 -1.1096692
```

```
> res$statistic

       tau          z
 0.3834711 -1.1096692

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.2671416

> res$p.value

[1] 0.2671416
```

- **Example 2:**

```
> x <- c(1, 2.3, 4.5, 6.7, 8.9)
> n <- length(x)
> rho <- sqrt((n - 1) * var(x)/n)
> tau <- mean(abs(x - mean(x)))
> res <- bonett.test(x, alternative = "two.sided")
> z <- sqrt(n + 2) * (13.29 * log(rho/tau) - 3)/3.54
> c(tau, z)

[1]  2.49600 -0.86214

> res$statistic

     tau        z
 2.49600 -0.86214

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.3886105

> res$p.value

[1] 0.3886105
```

- **Example 3:**

```
> x <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- length(x)
> rho <- sqrt((n - 1) * var(x)/n)
> tau <- mean(abs(x - mean(x)))
> res <- bonett.test(x, alternative = "two.sided")
> z <- sqrt(n + 2) * (13.29 * log(rho/tau) - 3)/3.54
> c(tau, z)

[1] 1.785000 1.035715

> res$statistic

     tau        z
1.785000 1.035715

> p.value <- 2 * pnorm(-abs(z))
> p.value

[1] 0.3003353

> res$p.value

[1] 0.3003353
```

12.2 Funzioni di adattamento normale

qqnorm()

- **Package:** stats
- **Input:**

  y vettore numerico di dimensione n ordinato in maniera crescente

  plot.it = FALSE
- **Description:** quantili teorici e campionari per QQ-Norm
- **Output:**

  x quantili teorici

  y quantili campionari
- **Formula:**

  x
$$\begin{cases} \Phi^{-1}\left((8\,i-3)\,/\,(8\,n+2)\right) & \forall\, i \,=\, 1,\, 2,\, \ldots,\, n \quad \text{se } n \leq 10 \\ \Phi^{-1}\left((i-1\,/\,2)\,/\,n\right) & \forall\, i \,=\, 1,\, 2,\, \ldots,\, n \quad \text{se } n > 10 \end{cases}$$

  y
$$y_{(i)} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Example 1:**

```
> y <- c(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 <- 7
> qqnorm(y, plot.it = FALSE)$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] -1.3644887 -0.7582926 -0.3529340  0.0000000  0.3529340  0.7582926  1.3644887

> qqnorm(y, plot.it = FALSE)$x

[1] -1.3644887 -0.7582926 -0.3529340  0.0000000  0.3529340  0.7582926  1.3644887
```

- **Example 2:**

```
> y <- c(1.2, 2.3, 4.3, -3.4, 4.2, 5.43, 3.2, 2.2, 0.2, 2.1, 2.2,
+     3.1)
> 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  4.30  5.43

> n <- 12
> qqnorm(y, plot = FALSE)$y

 [1] -3.40  0.20  1.20  2.10  2.20  2.20  2.30  3.10  3.20  4.20  4.30  5.43

> qnorm(((1:n) - 1/2)/n)
```

```
 [1] -1.7316644 -1.1503494 -0.8122178 -0.5485223 -0.3186394 -0.1046335
 [7]  0.1046335  0.3186394  0.5485223  0.8122178  1.1503494  1.7316644
```

```
> qqnorm(y, plot.it = FALSE)$x
```

```
 [1] -1.7316644 -1.1503494 -0.8122178 -0.5485223 -0.3186394 -0.1046335
 [7]  0.1046335  0.3186394  0.5485223  0.8122178  1.1503494  1.7316644
```

- **Example 3:**

```
> y <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- sort(y)
> y
```

```
[1] 1.1 2.3 3.4 4.5 5.6 6.7 6.7 8.9
```

```
> n <- 8
> qqnorm(y, plot.it = FALSE)$y
```

```
[1] 1.1 2.3 3.4 4.5 5.6 6.7 6.7 8.9
```

```
> qnorm((8 * (1:n) - 3)/(8 * n + 2))
```

```
[1] -1.4342002 -0.8524950 -0.4727891 -0.1525060  0.1525060  0.4727891  0.8524950
[8]  1.4342002
```

```
> qqnorm(y, plot.it = FALSE)$x
```

```
[1] -1.4342002 -0.8524950 -0.4727891 -0.1525060  0.1525060  0.4727891  0.8524950
[8]  1.4342002
```

ppoints()

- **Package:** stats
- **Input:**

  n valore naturale

- **Description:** rapporti per QQ-Norm
- **Formula:**

$$\begin{cases} (8\,i - 3)\,/\,(8\,n + 2) & \forall\, i\, =\, 1,\, 2,\, \ldots,\, n \quad \text{se } n \leq 10 \\ (i - 1\,/\,2)\,/\,n & \forall\, i\, =\, 1,\, 2,\, \ldots,\, n \quad \text{se } n > 10 \end{cases}$$

- **Example 1:**

```
> n <- 5
> (8 * (1:n) - 3)/(8 * n + 2)
```

```
[1] 0.1190476 0.3095238 0.5000000 0.6904762 0.8809524
```

```
> ppoints(n = 5)
```

```
[1] 0.1190476 0.3095238 0.5000000 0.6904762 0.8809524
```

- **Example 2:**

```
> n <- 12
> ((1:n) - 1/2)/n
```

```
 [1] 0.04166667 0.12500000 0.20833333 0.29166667 0.37500000 0.45833333
 [7] 0.54166667 0.62500000 0.70833333 0.79166667 0.87500000 0.95833333
```

```
> ppoints(n = 12)
```

```
 [1] 0.04166667 0.12500000 0.20833333 0.29166667 0.37500000 0.45833333
 [7] 0.54166667 0.62500000 0.70833333 0.79166667 0.87500000 0.95833333
```

- **Example 3:**

```
> n <- 15
> ((1:n) - 1/2)/n
```

```
 [1] 0.03333333 0.10000000 0.16666667 0.23333333 0.30000000 0.36666667
 [7] 0.43333333 0.50000000 0.56666667 0.63333333 0.70000000 0.76666667
[13] 0.83333333 0.90000000 0.96666667
```

```
> ppoints(n = 15)
```

```
 [1] 0.03333333 0.10000000 0.16666667 0.23333333 0.30000000 0.36666667
 [7] 0.43333333 0.50000000 0.56666667 0.63333333 0.70000000 0.76666667
[13] 0.83333333 0.90000000 0.96666667
```

12.3 Test di ipotesi su una distribuzione generica

Test Chi - Quadrato GOF

- **Package:** `stats`
- **Sintassi:** `chisq.test()`
- **Input:**

  `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`

$$df = k - 1$$

  `p.value`

$$P(\chi^2_{df} \geq c)$$

  `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`

$$\frac{n_i - \hat{n}_i}{\sqrt{\hat{n}_i}} \quad \forall\, i = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(100, 110, 80, 55, 14)
> n <- sum(x)
> n

[1] 359

> prob <- c(0.29, 0.21, 0.17, 0.17, 0.16)
> k <- 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, df = 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
```

- **Example 2:**

```
> x <- c(89, 37, 30, 28, 2)
> n <- sum(x)
> n

[1] 186

> prob <- c(0.4, 0.2, 0.2, 0.15, 0.05)
> k <- 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, df = 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
```

- **Example 3:**

```
> x <- c(54, 29, 5)
> n <- sum(x)
> n

[1] 88

> prob <- c(0.5, 0.25, 0.25)
> k <- 3
> osservati <- x
> attesi <- n * prob
> c <- sum((osservati - attesi)^2/attesi)
> c

[1] 17.63636

> chisq.test(x, p = prob)$statistic

X-squared
 17.63636

> parameter <- k - 1
> parameter

[1] 2

> chisq.test(x, p = prob)$parameter

df
 2

> p.value <- 1 - pchisq(c, df = parameter)
> p.value

[1] 0.0001480172

> chisq.test(x, p = prob)$p.value

[1] 0.0001480172
```

```
> osservati

[1] 54 29  5

> chisq.test(x, p = prob)$observed

[1] 54 29  5

> attesi

[1] 44 22 22

> chisq.test(x, p = prob)$expected

[1] 44 22 22

> residui <- (osservati - attesi)/sqrt(attesi)
> residui

[1]  1.507557  1.492405 -3.624412

> chisq.test(x, p = prob)$residuals

[1]  1.507557  1.492405 -3.624412
```

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: $RSS \,=\, \sum_{i=1}^n e_i^2 \,=\, y^T\,e \,=\, y^T\,(I_n - H)\,y$
- 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: $s_{-i}^2 \,=\, s^2 \left(1 + \frac{1 - rstandard_i^2}{n-3}\right) \,=\, s^2 \left(1 + \frac{rstudent_i^2 - 1}{n-2}\right)^{-1} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- codevianza tra x ed y: $ss_{xy} \,=\, \sum_{i=1}^n\,(x_i - \bar{x})\,(y_i - \bar{y})$
- devianza di x: $ss_x \,=\, \sum_{i=1}^n\,(x_i - \bar{x})^2$
- devianza di y: $ss_y \,=\, \sum_{i=1}^n\,(y_i - \bar{y})^2$
- stime OLS: $\hat{\beta} \,=\, (X^T\,X)^{-1}\,X^T\,y$
- stima OLS intercetta: $\hat{\beta}_1 \,=\, \bar{y} - \bar{x}\,ss_{xy}\,/\,ss_x$
- stima OLS coefficiente angolare: $\hat{\beta}_2 \,=\, ss_{xy}\,/\,ss_x$
- standard error delle stime OLS: $s_{\hat{\beta}} \,=\, s\,\sqrt{\text{diag}((X^T\,X)^{-1})}$
- standard error della stima OLS intercetta: $s_{\hat{\beta}_1} \,=\, s\,\sqrt{\sum_{i=1}^n x_i^2\,/\,(n\,ss_x)}$
- standard error della stima OLS coefficiente angolare: $s_{\hat{\beta}_2} \,=\, s\,/\,\sqrt{ss_x}$
- covarianza tra le stime OLS: $s_{\hat{\beta}_1\,\hat{\beta}_2} \,=\, -\bar{x}\,s^2\,/\,ss_x$
- 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: $rstudent_i \,=\, \frac{e_i}{s_{-i}\,\sqrt{1-h_i}} \,=\, rstandard_i\,\sqrt{\frac{n-3}{n-2-rstandard_i^2}} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- valori adattati: $\hat{y} \,=\, H\,y$

- valori di leva: $h_i = H_{i,i} \quad \forall\, i = 1, 2, \ldots, n$
- 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: $R^2 = 1 - RSS\,/\,RSS_{nullo} = 1 - (1 - R^2_{adj})\,(n-2)\,/\,(n-1) = r^2_{xy}$
- indice di determinazione aggiustato: $R^2_{adj} = 1 - \frac{RSS\,/\,(n-2)}{RSS_{nullo}\,/\,(n-1)} = 1 - \left(1 - R^2\right)\,(n-1)\,/\,(n-2)$
- valore noto del regressore per la previsione: x_0
- log-verosimiglianza normale: $\hat{\ell} = -n\,(\log(2\,\pi) + \log\,(RSS\,/\,n) + 1)\,/\,2$
- 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
- **Input:**

  `formula` modello di regressione lineare con una variabile esplicativa ed n unità

  `x = TRUE` matrice del modello

  `y = TRUE` variabile dipendente

- **Description:** analisi di regressione lineare
- **Output:**

  `coefficients` stime OLS

  `residuals` residui

  `rank` rango della matrice del modello

  `fitted.values` valori adattati

  `df.residual` gradi di libertà della devianza residua

  `x` matrice del modello

  `y` variabile dipendente

- **Formula:**

  `coefficients`
  $$\hat{\beta}_j \quad \forall\, j = 1, 2$$

  `residuals`
  $$e_i \quad \forall\, i = 1, 2, \ldots, n$$

  `rank`
  $$2$$

  `fitted.values`
  $$\hat{y}_i \quad \forall\, i = 1, 2, \ldots, n$$

  `df.residual`
  $$n - 2$$

  `x`
  $$X$$

  `y`
  $$y$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, x = TRUE, y = TRUE)
> modello$coefficients

(Intercept)           x
  3.8486818   0.7492486

> modello$residuals

          1           2           3           4           5           6
-3.17285530  0.82804637  2.37969944 -0.06864749 -1.65699442  1.40387291
          7           8
 0.55552598 -0.26864749

> modello$rank

[1] 2

> modello$fitted.values

        1         2         3         4         5         6         7         8
 4.672855  5.571954  7.220301  8.868647 10.516994  6.396127  8.044474  8.868647

> modello$df.residual

[1] 6

> modello$x

  (Intercept)   x
1           1 1.1
2           1 2.3
3           1 4.5
4           1 6.7
5           1 8.9
6           1 3.4
7           1 5.6
8           1 6.7
attr(,"assign")
[1] 0 1

> modello$y

   1    2    3    4    5    6    7    8
1.50 6.40 9.60 8.80 8.86 7.80 8.60 8.60
```

- **Note 1:** Il modello nullo si ottiene con `lm(formula = y ~ 1)`.
- **Note 2:** L'istruzione `lm(formula = y ~ x)` è equivalente a `lm(formula = y ~ X - 1)`.
- **Note 3:** L'istruzione `lm(formula = y ~ x)` è equivalente a `lm(formula = y ~ 1 + x)`.

summary.lm()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

  `correlation = TRUE` correlazione tra le stime OLS
- **Description:** 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`

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

  `coefficients`

$$\hat{\beta}_j \qquad s_{\hat{\beta}_j} \qquad t_{\hat{\beta}_j} \qquad p\text{-value} = 2\, P(t_{n-2} \leq -\,|\, t_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1, 2$$

  `sigma`

$$s$$

  `r.squared`

$$R^2$$

  `adj.r.squared`

$$R^2_{adj}$$

  `fstatistic`

$$F value = \frac{RSS_{nullo} - RSS}{RSS \,/\, (n-2)} = t^2_{\hat{\beta}_2} \qquad 1 \qquad n-2$$

  `cov.unscaled`

$$(X^T\, X)^{-1}$$

  `correlation`

$$r_{\hat{\beta}_1\, \hat{\beta}_2}$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> res <- summary.lm(object = modello, correlation = TRUE)
> res$residuals

          1           2           3           4           5           6
-3.17285530  0.82804637  2.37969944 -0.06864749 -1.65699442  1.40387291
          7           8
 0.55552598 -0.26864749

> res$coefficients
```

```
                Estimate Std. Error  t value   Pr(>|t|)
(Intercept) 3.8486818  1.5155372 2.539484 0.04411163
x           0.7492486  0.2774737 2.700251 0.03556412
```

```
> res$sigma
```

```
[1] 1.893745
```

```
> res$r.squared
```

```
[1] 0.5485788
```

```
> res$adj.r.squared
```

```
[1] 0.4733419
```

```
> res$fstatistic
```

```
   value    numdf    dendf
7.291356 1.000000 6.000000
```

```
> res$cov.unscaled
```

```
            (Intercept)           x
(Intercept)   0.6404573 -0.10519536
x            -0.1051954  0.02146844
```

```
> res$correlation
```

```
            (Intercept)          x
(Intercept)   1.0000000 -0.8971215
x            -0.8971215  1.0000000
```

vcov()

- **Package:** `stats`
- **Input:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** matrice di covarianza delle stime OLS
- **Formula:**

$$s^2\,(X^T\,X)^{-1}$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> vcov(object = modello)
```

```
            (Intercept)           x
(Intercept)   2.2968531 -0.37725904
x            -0.3772590  0.07699164
```

lm.fit()

- **Package:** stats
- **Input:**

  x matrice del modello

  y variabile dipendente

- **Description:** analisi di regressione lineare
- **Output:**

  coefficients stime OLS

  residuals residui

  rank rango della matrice del modello

  fitted.values valori adattati

  df.residual gradi di libertà della devianza residua

- **Formula:**

  coefficients
$$\hat{\beta}_j \quad \forall\, j = 1, 2$$
  residuals
$$e_i \quad \forall\, i = 1, 2, \ldots, n$$
  rank
$$2$$
  fitted.values
$$\hat{y}_i \quad \forall\, i = 1, 2, \ldots, n$$
  df.residual
$$n - 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> X <- model.matrix(object = modello)
> res <- lm.fit(x = X, y)
> res$coefficients

(Intercept)           x
  3.8486818   0.7492486

> res$residuals

[1] -3.17285530  0.82804637  2.37969944 -0.06864749 -1.65699442  1.40387291
[7]  0.55552598 -0.26864749

> res$rank

[1] 2

> res$fitted.values

[1]  4.672855  5.571954  7.220301  8.868647 10.516994  6.396127  8.044474
[8]  8.868647

> res$df.residual

[1] 6
```

lsfit()

- **Package:** stats
- **Input:**

  `x` matrice del modello

  `y` variabile dipendente

  `intercept = FALSE`
- **Description:** analisi di regressione lineare
- **Output:**

  `coefficients` stime OLS

  `residuals` residui
- **Formula:**

  `coefficients`
$$\hat{\beta}_j \quad \forall\, j = 1, 2$$
  `residuals`
$$e_i \quad \forall\, i = 1, 2, \ldots, n$$
- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> X <- model.matrix(object = modello)
> res <- lsfit(x = X, y, intercept = FALSE)
> res$coefficients

(Intercept)           x
  3.8486818   0.7492486

> res$residuals

[1] -3.17285530  0.82804637  2.37969944 -0.06864749 -1.65699442  1.40387291
[7]  0.55552598 -0.26864749
```

confint()

- **Package:** stats
- **Input:**

  `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 $1 - \alpha$
- **Description:** intervallo di confidenza per le stime OLS
- **Formula:**
$$\hat{\beta}_j \mp t_{1-\alpha\,/\,2,\, n-2}\, s_{\hat{\beta}_j} \quad \forall j = 1, 2$$
- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> confint(object = modello, parm = c(1, 2), level = 0.95)
```

```
                 2.5 %   97.5 %
(Intercept) 0.14029581 7.557068
x           0.07029498 1.428202
```

coef()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** stime OLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> coef(object = modello)

(Intercept)           x
  3.8486818   0.7492486
```

boxcox()

- **Package:** MASS
- **Input:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

  `lambda` parametro di trasformazione λ

  `plotit = FALSE`

- **Description:** 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.

- **Example 1:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> res <- boxcox(object = modello, lambda = 1.2, plotit = FALSE)
> res$x
```

```
[1] 1.2
```

```
> res$y
```

```
[1] -11.69470
```

- **Example 2:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> res <- boxcox(object = modello, lambda = 4.1, plotit = FALSE)
> res$x
```

```
[1] 4.1
```

```
> res$y
```

```
[1] -11.30996
```

fitted()

- **Package:** stats
- **Input:**

  object modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{y}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> fitted(object = modello)
```

```
       1         2         3         4         5         6         7         8 
4.672855  5.571954  7.220301  8.868647 10.516994  6.396127  8.044474  8.868647 
```

predict.lm()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` 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 $1 - \alpha$

- **Description:** 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"`

$$\hat{\beta}_1 + \hat{\beta}_2\, x_0 \qquad \hat{\beta}_1 + \hat{\beta}_2\, x_0 \mp t_{1-\alpha\,/\,2,\,df}\; s^* \sqrt{\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}}$$

  `interval = "prediction"`

$$\hat{\beta}_1 + \hat{\beta}_2\, x_0 \qquad \hat{\beta}_1 + \hat{\beta}_2\, x_0 \mp t_{1-\alpha\,/\,2,\,df}\; s^* \sqrt{1 + \frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_{i=1}^n (x_i - \bar{x})^2}}$$

  `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`

$$s^*$$

- **Example 1:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> x0 <- c(1, 1.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat

[1] 4.822705

> new <- data.frame(x = 1.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> lower <- yhat - qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     X) %*% x0)
> upper <- yhat + qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     X) %*% x0)
> c(yhat, lower, upper)
```

```
[1] 4.822705 2.465776 7.179634

> res <- predict.lm(object = modello, newdata = new, se.fit = TRUE,
+     scale = s, df = Inf, interval = "confidence", level = 0.95)
> res$fit

       fit      lwr      upr
1 4.822705 2.465776 7.179634

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% X) %*%
+     x0))
> se.fit

[1] 1.202537

> res$se.fit

[1] 1.202537

> s

[1] 1.893745

> res$residual.scale

[1] 1.893745
```

- **Example 2:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> x0 <- c(1, 1.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat

[1] 4.822705

> new <- data.frame(x = 1.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> lower <- yhat - qt(1 - 0.05/2, df = n - 2) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% X) %*% x0)
> upper <- yhat + qt(1 - 0.05/2, df = n - 2) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% X) %*% x0)
> c(yhat, lower, upper)

[1]  4.8227050 -0.6664366 10.3118467

> res <- predict.lm(object = modello, newdata = new, se.fit = TRUE,
+     interval = "prediction", level = 0.95)
> res$fit

       fit        lwr      upr
1 4.822705 -0.6664366 10.31185

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% X) %*%
+     x0))
> se.fit
```

```
[1] 1.202537
> res$se.fit
[1] 1.202537
> s
[1] 1.893745
> res$residual.scale
[1] 1.893745
```

- **Note 1:** Per il calcolo dell'intervallo classico di confidenza o previsione impostare i parametri `df = n - 2` e `scale = summary.lm(object = modello)$sigma`.
- **Note 2:** Per il calcolo dell'intervallo asintotico di confidenza o previsione impostare i parametri `df = Inf` e `scale = summary.lm(object = modello)$sigma`.

predict()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` 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 $1 - \alpha$
- **Description:** 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"`

$$\hat{\beta}_1 + \hat{\beta}_2 \, x_0 \qquad \hat{\beta}_1 + \hat{\beta}_2 \, x_0 \mp t_{1-\alpha\,/\,2,\,df} \, s^* \sqrt{\frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}$$

  `interval = "prediction"`

$$\hat{\beta}_1 + \hat{\beta}_2 \, x_0 \qquad \hat{\beta}_1 + \hat{\beta}_2 \, x_0 \mp t_{1-\alpha\,/\,2,\,df} \, s^* \sqrt{1 + \frac{1}{n} + \frac{(x_0 - \bar{x})^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2}}$$

  `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

$$s^*$$

- **Example 1:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> x0 <- c(1, 1.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat

[1] 4.822705

> new <- data.frame(x = 1.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> lower <- yhat - qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     X) %*% x0)
> upper <- yhat + qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     X) %*% x0)
> c(yhat, lower, upper)

[1] 4.822705 2.465776 7.179634

> res <- predict(object = modello, newdata = new, se.fit = TRUE,
+     scale = s, df = Inf, interval = "confidence", level = 0.95)
> res$fit

       fit      lwr      upr
1 4.822705 2.465776 7.179634

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% X) %*%
+     x0))
> se.fit

[1] 1.202537

> res$se.fit

[1] 1.202537

> s

[1] 1.893745

> res$residual.scale

[1] 1.893745
```

- **Example 2:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> x0 <- c(1, 1.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat
```

```
[1] 4.822705

> new <- data.frame(x = 1.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> lower <- yhat - qt(1 - 0.05/2, df = n - 2) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% X) %*% x0)
> upper <- yhat + qt(1 - 0.05/2, df = n - 2) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% X) %*% x0)
> c(yhat, lower, upper)

[1]  4.8227050 -0.6664366 10.3118467

> res <- predict(object = modello, newdata = new, se.fit = TRUE,
+     interval = "prediction", level = 0.95)
> res$fit

       fit        lwr      upr
1 4.822705 -0.6664366 10.31185

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% X) %*%
+     x0))
> se.fit

[1] 1.202537

> res$se.fit

[1] 1.202537

> s

[1] 1.893745

> res$residual.scale

[1] 1.893745
```

- **Note 1:** Per il calcolo dell'intervallo classico di confidenza o previsione impostare i parametri `df = n - 2` e `scale = summary.lm(object = modello)$sigma`.
- **Note 2:** Per il calcolo dell'intervallo asintotico di confidenza o previsione impostare i parametri `df = Inf` e `scale = summary.lm(object = modello)$sigma`.

cov2cor()

- **Package:** stats
- **Input:**

  `V` matrice di covarianza delle stime OLS di dimensione 2×2

- **Description:** converte la matrice di covarianza nella matrice di correlazione
- **Formula:**

$$r_{\hat{\beta}_i \, \hat{\beta}_j} \quad \forall \, i, j \, = \, 1, \, 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> V <- vcov(object = modello)
> cov2cor(V)
```

```
            (Intercept)          x
(Intercept)   1.0000000 -0.8971215
x            -0.8971215  1.0000000
```

13.3 Adattamento

logLik()

- **Package:** stats
- **Input:**

  object modello di regressione lineare con una variabile esplicativa ed n unità
- **Description:** log-verosimiglianza normale
- **Formula:**

$$\hat{\ell}$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> logLik(object = modello)
```

```
'log Lik.' -15.30923 (df=3)
```

durbin.watson()

- **Package:** car
- **Input:**

  model modello di regressione lineare con una variabile esplicativa ed n unità
- **Description:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

  dw valore empirico della statistica *D–W*
- **Formula:**

  dw

$$\sum_{i=2}^{n} (e_i - e_{i-1})^2 \,/\, RSS$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> durbin.watson(model = modello)$dw
```

```
[1] 1.75205
```

AIC()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità
- **Description:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell} + 6$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> AIC(object = modello)

[1] 36.61846
```

extractAIC()

- **Package:** stats
- **Input:**

  `fit` modello di regressione lineare con una variabile esplicativa ed n unità
- **Description:** numero di parametri del modello ed indice *AIC* generalizzato
- **Formula:**

$$2 \qquad n\,\log(RSS\,/\,n)\,+4$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> extractAIC(fit = modello)

[1]  2.00000 11.91545
```

deviance()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità
- **Description:** devianza residua
- **Formula:**

$$RSS$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> deviance(object = modello)

[1] 21.51762
```

PRESS()

- **Package:** MPV
- **Input:**

  `x` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** PRESS
- **Formula:**

$$\sum_{i=1}^{n} e_i^2 \,/\, (1 - h_i)^2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> PRESS(x = modello)

[1] 53.41271
```

anova()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** anova di regressione
- **Output:**

  `Df` gradi di libertà

  `Sum Sq` devianze residue

  `Mean Sq` quadrati medi

  `F value` valore empirico della statistica F

  `Pr(>F)` p-value

- **Formula:**

  `Df`

$$1 \qquad n-2$$

  `Sum Sq`

$$RSS_{nullo} - RSS \qquad RSS$$

  `Mean Sq`

$$RSS_{nullo} - RSS \qquad RSS \,/\, (n-2)$$

  `F value`

$$F_{value} = \frac{RSS_{nullo} - RSS}{RSS \,/\, (n-2)} = t^2_{\hat{\beta}_2}$$

  `Pr(>F)`

$$P(F_{1,\,n-2} \geq F_{value})$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> anova(object = modello)
```

```
Analysis of Variance Table

Response: y
          Df  Sum Sq Mean Sq F value  Pr(>F)
x          1 26.1488 26.1488  7.2914 0.03556 *
Residuals  6 21.5176  3.5863
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

drop1()

- **Package:** `stats`
- **Input:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

  `scale` selezione indice AIC oppure Cp

  `test = "F"`
- **Description:** 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`

$$RSS_{nullo} - RSS$$

  `RSS`

$$RSS,\; RSS_{nullo}$$

  `AIC`

`scale = 0`

$$n\,\log\left(RSS\,/\,n\right) + 4,\; n\,\log\left(RSS_{nullo}\,/\,n\right) + 2$$

  `Cp`

`scale =` s^2

$$2,\; \frac{RSS_{nullo}}{RSS\,/\,(n-2)} + 2 - n$$

  `F value`

$$F_{value} = \frac{RSS_{nullo} - RSS}{RSS\,/\,(n-2)} = t^2_{\hat{\beta}_2}$$

  `Pr(F)`

$$P(F_{1,\,n-2} \geq F_{value})$$

- **Example 1:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> drop1(object = modello, scale = 0, test = "F")

Single term deletions

Model:
y ~ x
       Df Sum of Sq    RSS    AIC F value   Pr(F)
<none>              21.518 11.915
x       1    26.149 47.666 16.278  7.2914 0.03556 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- **Example 2:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> s <- summary.lm(object = modello)$sigma
> drop1(object = modello, scale = s^2, test = "F")

Single term deletions

Model:
y ~ x

scale:  3.586271

       Df Sum of Sq    RSS     Cp F value   Pr(F)
<none>              21.518 2.0000
x       1    26.149 47.666 7.2914  7.2914 0.03556 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

add1()

- **Package:** `stats`
- **Input:**

  `object` modello nullo di regressione lineare semplice

  `scope` modello di regressione lineare con una variabile esplicativa ed n unità

  `scale` selezione indice AIC oppure Cp

  `test = "F"`
- **Description:** 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`

  $$RSS_{nullo} - RSS$$

  `RSS`

  $$RSS_{nullo},\, RSS$$

  `AIC`

  `scale = 0`

  $$n\,\log\left(RSS_{nullo}\,/\,n\right) + 2,\; n\,\log\left(RSS\,/\,n\right) + 4$$

  `Cp`

  `scale =` s^2

  $$\frac{RSS_{nullo}}{RSS\,/\,(n-2)} + 2 - n,\; 2$$

  `F value`

  $$F_{value} = \frac{RSS_{nullo} - RSS}{RSS\,/\,(n-2)} = t^2_{\hat\beta_2}$$

  `Pr(F)`

  $$P(F_{1,\,n-2} \geq F_{value})$$

- **Example 1:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> nullo <- lm(formula = y ~ 1)
> add1(object = nullo, scope = modello, scale = 0, test = "F")

Single term additions

Model:
y ~ 1
       Df Sum of Sq    RSS    AIC F value   Pr(F)
<none>               47.666 16.278
x       1    26.149 21.518 11.915  7.2914 0.03556 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- **Example 2:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> nullo <- lm(formula = y ~ 1)
> s <- summary.lm(object = modello)$sigma
> add1(object = nullo, scope = modello, scale = s^2, test = "F")

Single term additions

Model:
y ~ 1

scale:  3.586271

       Df Sum of Sq    RSS     Cp F value   Pr(F)
```

```
<none>              47.666 7.2914
x        1    26.149 21.518 2.0000  7.2914 0.03556 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

13.4 Diagnostica

ls.diag()

- **Package:** stats
- **Input:**

  ls.out modello di regressione lineare con una variabile eplicativa ed n unità

- **Description:** 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

  $$h_i \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

  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

  $$s_{\hat{\beta}_j} \quad \forall\, j \,=\, 1,\, 2$$

  cov.scaled

  $$s^2\, (X^T\, X)^{-1}$$

  cov.unscaled

  $$(X^T\, X)^{-1}$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> res <- ls.diag(ls.out = modello)
> res$std.dev

[1] 1.893745

> res$hat

[1] 0.4350043 0.2701267 0.1284350 0.1945578 0.4684951 0.1733040 0.1355195
[8] 0.1945578

> res$std.res

[1] -2.22897996  0.51181072  1.34601741 -0.04039112 -1.20017856  0.81532985
[7]  0.31550428 -0.15806803

> res$stud.res

[1] -4.90710471  0.47776268  1.47068630 -0.03687690 -1.25680777  0.78929887
[7]  0.29043398 -0.14459710

> res$cooks

[1] 1.9126289653 0.0484739848 0.1334918569 0.0001970407 0.6348329327
[6] 0.0696786009 0.0078023824 0.0030176734

> res$dfits

[1] -4.30575707  0.29065126  0.56456215 -0.01812431 -1.17996116  0.36138726
[7]  0.11499284 -0.07106678

> res$correlation

            (Intercept)          x
(Intercept)   1.0000000 -0.8971215
x            -0.8971215  1.0000000

> res$std.err

                 [,1]
(Intercept) 1.5155372
x           0.2774737

> res$cov.scaled

            (Intercept)           x
(Intercept)   2.2968531 -0.37725904
x            -0.3772590  0.07699164

> res$cov.unscaled

            (Intercept)           x
(Intercept)   0.6404573 -0.10519536
x            -0.1051954  0.02146844
```

cooks.distance()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> cooks.distance(model = modello)

           1            2            3            4            5            6
1.9126289653 0.0484739848 0.1334918569 0.0001970407 0.6348329327 0.0696786009
           7            8
0.0078023824 0.0030176734
```

rstandard()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> rstandard(model = modello)

          1           2           3           4           5           6
-2.22897996  0.51181072  1.34601741 -0.04039112 -1.20017856  0.81532985
          7           8
 0.31550428 -0.15806803
```

rstandard.lm()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> rstandard.lm(model = modello)

          1           2           3           4           5           6
-2.22897996  0.51181072  1.34601741 -0.04039112 -1.20017856  0.81532985
          7           8
 0.31550428 -0.15806803
```

rstudent()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> rstudent(model = modello)

          1           2           3           4           5           6
-4.90710471  0.47776268  1.47068630 -0.03687690 -1.25680777  0.78929887
          7           8
 0.29043398 -0.14459710
```

rstudent.lm()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> rstudent.lm(model = modello)

          1           2           3           4           5           6
-4.90710471  0.47776268  1.47068630 -0.03687690 -1.25680777  0.78929887
          7           8
 0.29043398 -0.14459710
```

lmwork()

- **Package:** MASS
- **Input:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** 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`

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> res <- lmwork(object = modello)
> res$stdedv

[1] 1.893745

> res$stdres

          1           2           3           4           5           6
-2.22897996  0.51181072  1.34601741 -0.04039112 -1.20017856  0.81532985
          7           8
 0.31550428 -0.15806803

> res$studres

          1           2           3           4           5           6
-4.90710471  0.47776268  1.47068630 -0.03687690 -1.25680777  0.78929887
          7           8
 0.29043398 -0.14459710
```

dffits()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** dffits
- **Formula:**

$$rstudent_i \sqrt{\frac{h_i}{1 - h_i}} \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> dffits(model = modello)

          1           2           3           4           5           6
-4.30575707  0.29065126  0.56456215 -0.01812431 -1.17996116  0.36138726
          7           8
 0.11499284 -0.07106678
```

covratio()

- **Package:** stats
- **Input:**

  model modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** covratio
- **Formula:**

$$cr_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> covratio(model = modello)

         1          2          3          4          5          6          7
0.07534912 1.80443448 0.80504974 1.78686556 1.56459066 1.37727804 1.61092794
         8
1.77297867
```

lm.influence()

- **Package:** stats
- **Input:**

  model modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** 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

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

  coefficients

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = e_i\,(1 - h_i)^{-1}\,(X^T\,X)^{-1}_j\,X_i^T \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2$$

sigma

$$s_{-i} \quad \forall i = 1, 2, \ldots, n$$

wt.res

$$e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> res <- lm.influence(model = modello)
> res$hat

        1         2         3         4         5         6         7         8
0.4350043 0.2701267 0.1284350 0.1945578 0.4684951 0.1733040 0.1355195 0.1945578

> res$coefficients

   (Intercept)            x
1 -2.946804056  0.458130527
2  0.452110031 -0.063325849
3  0.456185994 -0.023446758
4  0.005484663 -0.003293542
5  0.922114131 -0.267715952
6  0.480231536 -0.054685694
7  0.033006665  0.009657123
8  0.021463873 -0.012889065

> res$sigma

        1         2         3         4         5         6         7         8
0.8602058 2.0287040 1.7332139 2.0742118 1.8084168 1.9562006 2.0572134 2.0701700

> res$wt.res

          1           2           3           4           5           6
-3.17285530  0.82804637  2.37969944 -0.06864749 -1.65699442  1.40387291
          7           8
 0.55552598 -0.26864749
```

residuals.lm()

- **Package:** stats
- **Input:**

  object modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** residui
- **Formula:**

$$e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> residuals.lm(object = modello)
```

```
          1           2           3           4           5           6
-3.17285530  0.82804637  2.37969944 -0.06864749 -1.65699442  1.40387291
          7           8
 0.55552598 -0.26864749
```

df.residual()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** gradi di libertà della devianza residua
- **Formula:**

$$n - 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> df.residual(object = modello)
```

```
[1] 6
```

hatvalues()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> hatvalues(model = modello)
```

```
        1         2         3         4         5         6         7         8
0.4350043 0.2701267 0.1284350 0.1945578 0.4684951 0.1733040 0.1355195 0.1945578
```

dfbeta()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** dfbeta
- **Formula:**

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = e_i\,(1 - h_i)^{-1}\,(X^T\,X)^{-1}_j\,X^T_i \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> dfbeta(model = modello)

   (Intercept)            x
1 -2.946804056  0.458130527
2  0.452110031 -0.063325849
3  0.456185994 -0.023446758
4  0.005484663 -0.003293542
5  0.922114131 -0.267715952
6  0.480231536 -0.054685694
7  0.033006665  0.009657123
8  0.021463873 -0.012889065
```

dfbetas()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** dfbetas
- **Formula:**

$$\frac{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}{s_{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}} = \frac{e_i\,(1 - h_i)^{-1}\,(X^T\,X)^{-1}_j\,X^T_i}{s_{-i}\,\sqrt{(X^T\,X)^{-1}_{j,\,j}}} \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> dfbetas(model = modello)

   (Intercept)           x
1 -4.280591734  3.63485094
2  0.278471258 -0.21304046
3  0.328885485 -0.09232735
4  0.003304089 -0.01083702
5  0.637149075 -1.01035839
6  0.306755388 -0.19079196
7  0.020048284  0.03203820
8  0.012955584 -0.04249278
```

outlier.test()

- **Package:**
- **Input:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** test sugli *outliers*
- **Output:**

  `test` massimo residuo studentizzato assoluto, gradi di libertà, p-value

- **Formula:**

  `test`

  $$t = \max_i(\,|\,rstudent_i\,|) \quad n-3 \quad p\text{-value} = 2\,P(\,t_{n-3} \leq -|\,t\,|) \qquad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x)
> res <- outlier.test(model = modello)
> res$test

max|rstudent|            df  unadjusted p  Bonferroni p
  4.907104708   5.000000000   0.004446945   0.035575564
```

influence.measures()

- **Package:** `stats`
- **Input:**

  `model` modello di regressione lineare con una variabile esplicativa ed n unità

- **Description:** dfbetas, dffits, covratio, distanza di *Cook*, valori di leva
- **Output:**

  `infmat` misure di influenza di dimensione $n \times 6$

  `is.inf` matrice di influenza con valori logici di dimensione $n \times 6$

- **Formula:**

  `infmat`

  $$DFBETAS_{ij} = \frac{e_i\,(1-h_i)^{-1}\,(X^T\,X)_j^{-1}\,X_i^T}{s_{-i}\,\sqrt{(X^T\,X)_{j,\,j}^{-1}}} \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2$$

  $$DFFITS_i = rstudent_i\,\sqrt{\frac{h_i}{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$$

  $$COVRATIO_i = (1-h_i)^{-1}\left(1 + \frac{rstudent_i^2 - 1}{n-2}\right)^{-2} \quad \forall i = 1, 2, \ldots, n$$

  $$COOKD_i = \frac{h_i\,rstandard_i^2}{2\,(1-h_i)} \quad \forall\, i = 1, 2, \ldots, n$$

  $$HAT_i = h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
```

```
> modello <- lm(formula = y ~ x)
> res <- influence.measures(model = modello)
> res

Influence measures of
         lm(formula = y ~ x) :

    dfb.1_   dfb.x   dffit  cov.r   cook.d   hat  inf
1 -4.28059  3.6349 -4.3058 0.0753 1.912629 0.435    *
2  0.27847 -0.2130  0.2907 1.8044 0.048474 0.270
3  0.32889 -0.0923  0.5646 0.8050 0.133492 0.128
4  0.00330 -0.0108 -0.0181 1.7869 0.000197 0.195
5  0.63715 -1.0104 -1.1800 1.5646 0.634833 0.468    *
6  0.30676 -0.1908  0.3614 1.3773 0.069679 0.173
7  0.02005  0.0320  0.1150 1.6109 0.007802 0.136
8  0.01296 -0.0425 -0.0711 1.7730 0.003018 0.195

> res$infmat

        dfb.1_        dfb.x        dffit       cov.r        cook.d       hat
1 -4.280591734  3.63485094 -4.30575707 0.07534912 1.9126289653 0.4350043
2  0.278471258 -0.21304046  0.29065126 1.80443448 0.0484739848 0.2701267
3  0.328885485 -0.09232735  0.56456215 0.80504974 0.1334918569 0.1284350
4  0.003304089 -0.01083702 -0.01812431 1.78686556 0.0001970407 0.1945578
5  0.637149075 -1.01035839 -1.17996116 1.56459066 0.6348329327 0.4684951
6  0.306755388 -0.19079196  0.36138726 1.37727804 0.0696786009 0.1733040
7  0.020048284  0.03203820  0.11499284 1.61092794 0.0078023824 0.1355195
8  0.012955584 -0.04249278 -0.07106678 1.77297867 0.0030176734 0.1945578

> res$is.inf

  dfb.1_ dfb.x dffit cov.r cook.d   hat
1   TRUE  TRUE  TRUE FALSE   TRUE FALSE
2  FALSE FALSE FALSE FALSE  FALSE FALSE
3  FALSE FALSE FALSE FALSE  FALSE FALSE
4  FALSE FALSE FALSE FALSE  FALSE FALSE
5  FALSE  TRUE FALSE FALSE  FALSE FALSE
6  FALSE FALSE FALSE FALSE  FALSE FALSE
7  FALSE FALSE FALSE FALSE  FALSE FALSE
8  FALSE FALSE FALSE FALSE  FALSE FALSE
```

- **Note 1:** Il caso i-esimo è influente se $|DFBETAS_{ij}| > 1 \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2$
- **Note 2:** Il caso i-esimo è influente se $|DFFITS_i| > 3 \sqrt{2 \,/\, (n-2)} \quad \forall i = 1, 2, \ldots, n$
- **Note 3:** Il caso i-esimo è influente se $|1 - COVRATIO_i| > 6 \,/\, (n-2) \quad \forall i = 1, 2, \ldots, n$
- **Note 4:** Il caso i-esimo è influente se $P\,(F_{2,n-2} \geq COOKD_i) > 0.5 \quad \forall i = 1, 2, \ldots, n$
- **Note 5:** Il caso i-esimo è influente se $HAT_i > 6 \,/\, n \quad \forall i = 1, 2, \ldots, n$
- **Note 6:** I casi influenti rispetto ad almeno una tra queste misure sono marcati con un asterisco. Corrispondentemente la stessa riga della matrice `is.inf` riporterà almeno un simbolo `TRUE`.

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: $RSS \,=\, \sum_{i=1}^{n} e_i^2 \,=\, y^T\,e \,=\, y^T\,(I_n - H)\,y$
- 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: $s_{-i}^2 \,=\, s^2 \left(1 + \frac{1 - rstandard_i^2}{n-k-1}\right) \,=\, s^2 \left(1 + \frac{rstudent_i^2 - 1}{n-k}\right)^{-1} \quad \forall\, i \,=\, 1, 2, \ldots, n$
- stime OLS: $\hat{\beta} \,=\, (X^T\,X)^{-1}\,X^T\,y$
- standard error delle stime OLS: $s_{\hat{\beta}} \,=\, s\,\sqrt{\text{diag}((X^T\,X)^{-1})}$
- 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: $rstudent_i \,=\, \frac{e_i}{s_{-i}\,\sqrt{1-h_i}} \,=\, rstandard_i\,\sqrt{\frac{n-k-1}{n-k-rstandard_i^2}} \quad \forall\, i \,=\, 1, 2, \ldots, n$
- valori adattati: $\hat{y} \,=\, H\,y$
- valori di leva: $h_i \,=\, H_{i,i} \quad \forall\, i \,=\, 1, 2, \ldots, n$
- 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: $R^2 \,=\, 1 - RSS\,/\,RSS_{nullo} \,=\, 1 - (1 - R^2_{adj})\,(n-k)\,/\,(n-1)$
- indice di determinazione aggiustato: $R^2_{adj} \,=\, 1 - \frac{RSS\,/\,(n-k)}{RSS_{nullo}\,/\,(n-1)} \,=\, 1 - \left(1 - R^2\right)\,(n-1)\,/\,(n-k)$
- valore noto dei regressori per la previsione: $x_0^T \,=\, (1, x_{01}, x_{02}, \ldots, x_{0k-1})$
- log-verosimiglianza normale: $\hat{\ell} \,=\, -n\,(\log(2\,\pi) + \log\,(RSS\,/\,n) + 1)\,/\,2$

- 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
- **Input:**

  `formula` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `x = TRUE` matrice del modello

  `y = TRUE` variabile dipendente

- **Description:** analisi di regressione lineare
- **Output:**

  `coefficients` stime OLS

  `residuals` residui

  `rank` rango della matrice del modello

  `fitted.values` valori adattati

  `df.residual` gradi di libertà della devianza residua

  `x` matrice del modello

  `y` variabile dipendente

- **Formula:**

  `coefficients`
  $$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

  `residuals`
  $$e_i \quad \forall\, i = 1, 2, \ldots, n$$

  `rank`
  $$k$$

  `fitted.values`
  $$\hat{y}_i \quad \forall\, i = 1, 2, \ldots, n$$

  `df.residual`
  $$n-k$$

  `x`
  $$X$$

  `y`
  $$y$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, x = TRUE, y = TRUE)
> modello$coefficients

 (Intercept)          x1           x2           x3
 0.988514333  0.422516384 -0.001737381  0.716029046

> modello$residuals
```

```
         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227
```

```
> modello$rank
```

```
[1] 4
```

```
> modello$fitted.values
```

```
        1         2         3         4         5         6         7         8
 2.453638  5.964158  8.293288  8.102518  8.602437  7.139221  9.569117 10.035623
```

```
> modello$df.residual
```

```
[1] 4
```

```
> modello$x
```

```
  (Intercept)  x1  x2   x3
1           1 1.1 1.2 1.40
2           1 2.3 3.4 5.60
3           1 4.5 5.6 7.56
4           1 6.7 7.5 6.00
5           1 8.9 7.5 5.40
6           1 3.4 6.7 6.60
7           1 5.6 8.6 8.70
8           1 6.7 7.6 8.70
attr(,"assign")
[1] 0 1 2 3
```

```
> modello$y
```

```
   1    2    3    4    5    6    7    8
1.50 6.40 9.60 8.80 8.86 7.80 8.60 8.60
```

- **Note 1:** Il modello nullo si ottiene con `lm(formula = y ~ 1)`.
- **Note 2:** L'istruzione `update(object = y ~ x1 + x2,formula = . ~ . + x3)` è esattamente equivalente a `lm(formula = y ~ x1 + x2 + x3)`.
- **Note 3:** In seguito ad una modifica come ad esempio `x1[3] <- 1.2`, conviene adoperare il comando `update(modello)` anziché ripetere `modello <- lm(formula = y ~ x1 + x2 + x3)` .
- **Note 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))`.
- **Note 5:** Per regressioni polinomiali occorre usare il comando `poly()`. Per un polinomio di quarto grado occorre scrivere `lm(formula = y ~ poly(x,degree = 4,raw = TRUE))`.
- **Note 6:** 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))`.
- **Note 7:** Il comando `lm(formula = y ~ x1 + x2)` è equivalente a `lm(formula = y ~ X-1)`.
- **Note 8:** Il comando `lm(formula = y ~ x1 + x2)` è equivalente a `lm(formula = y ~ 1 + x1 + x2)`.

summary.lm()

- **Package:** stats
- **Input:**

  object modello di regressione lineare con $k - 1$ variabili esplicative ed n unità

  correlation = TRUE correlazione tra le stime OLS

- **Description:** 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

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

  coefficients

$$\hat{\beta}_j \qquad s_{\hat{\beta}_j} \qquad t_{\hat{\beta}_j} \qquad p\text{-value} = 2\,P(t_{n-k} \le -\,|\,t_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1, 2, \ldots, k$$

  sigma

$$s$$

  r.squared

$$R^2$$

  adj.r.squared

$$R^2_{adj}$$

  fstatistic

$$F value = \frac{(RSS_{nullo} - RSS)\,/\,(k-1)}{RSS\,/\,(n-k)} \qquad k-1 \qquad n-k$$

  cov.unscaled

$$(X^T\,X)^{-1}$$

  correlation

$$r_{\hat{\beta}_i\,\hat{\beta}_j} \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> res <- summary.lm(object = modello, correlation = TRUE)
> res$residuals

         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227
```

```
> res$coefficients

               Estimate Std. Error      t value  Pr(>|t|)
(Intercept)  0.988514333  1.4292308  0.691640822 0.5272118
x1           0.422516384  0.3883267  1.088043731 0.3377443
x2          -0.001737381  0.5822146 -0.002984091 0.9977619
x3           0.716029046  0.4068987  1.759723294 0.1532663

> res$sigma

[1] 1.303508

> res$r.squared

[1] 0.8574147

> res$adj.r.squared

[1] 0.7504757

> res$fstatistic

   value    numdf    dendf
8.017793 3.000000 4.000000

> res$cov.unscaled

            (Intercept)          x1         x2          x3
(Intercept)  1.20220217 -0.06075872  0.0350553 -0.15856757
x1          -0.06075872  0.08874976 -0.1093953  0.04541621
x2           0.03505530 -0.10939532  0.1994982 -0.11184964
x3          -0.15856757  0.04541621 -0.1118496  0.09744180

> res$correlation

            (Intercept)         x1          x2         x3
(Intercept)  1.00000000 -0.1860100  0.07158062 -0.4632900
x1          -0.18600997  1.0000000 -0.82213982  0.4883764
x2           0.07158062 -0.8221398  1.00000000 -0.8022181
x3          -0.46329002  0.4883764 -0.80221810  1.0000000
```

vcov()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Description:** matrice di covarianza delle stime OLS
- **Formula:**

$$s^2 \, (X^T \, X)^{-1}$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> vcov(object = modello)

            (Intercept)          x1          x2          x3
(Intercept)  2.04270054 -0.10323710  0.05956359 -0.26942727
x1          -0.10323710  0.15079759 -0.18587712  0.07716815
x2           0.05956359 -0.18587712  0.33897378 -0.19004733
x3          -0.26942727  0.07716815 -0.19004733  0.16556652
```

lm.fit()

- **Package:** stats
- **Input:**

  x matrice del modello

  y variabile dipendente

- **Description:** analisi di regressione lineare
- **Output:**

  coefficients stime OLS

  residuals residui

  rank rango della matrice del modello

  fitted.values valori adattati

  df.residual gradi di libertà della devianza residua

- **Formula:**

  coefficients
  $$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

  residuals
  $$e_i \quad \forall\, i = 1, 2, \ldots, n$$

  rank
  $$k$$

  fitted.values
  $$\hat{y}_i \quad \forall\, i = 1, 2, \ldots, n$$

  df.residual
  $$n - k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> X <- model.matrix(object = modello)
> res <- lm.fit(x = X, y)
> res$coefficients
```

```
 (Intercept)          x1           x2           x3
 0.988514333  0.422516384 -0.001737381  0.716029046
```

```
> res$residuals
```

```
[1] -0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
[8] -1.4356227
```

```
> res$rank
```

```
[1] 4
```

```
> res$fitted.values
```

```
[1]  2.453638  5.964158  8.293288  8.102518  8.602437  7.139221  9.569117
[8] 10.035623
```

```
> res$df.residual
```

```
[1] 4
```

lsfit()

- **Package:** stats
- **Input:**

  x matrice del modello

  y variabile dipendente

  intercept = FALSE

- **Description:** analisi di regressione lineare
- **Output:**

  coefficients stime OLS

  residuals residui

- **Formula:**

  coefficients

  $$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

  residuals

  $$e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> X <- model.matrix(object = modello)
> res <- lsfit(x = X, y, intercept = FALSE)
> res$coefficients
```

```
 (Intercept)          x1           x2           x3
 0.988514333  0.422516384 -0.001737381  0.716029046
```

```
> res$residuals
```

```
[1] -0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
[8] -1.4356227
```

confint()

- **Package:** stats
- **Input:**

  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 $1-\alpha$

- **Description:** intervallo di confidenza per le stime OLS
- **Formula:**

$$\hat{\beta}_j \mp t_{1-\alpha\,/\,2,\,n-k}\, s_{\hat{\beta}_j} \quad \forall j = 1, 2, \ldots, k$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> confint(object = modello, parm = c(1, 2, 3, 4), level = 0.95)

                 2.5 %   97.5 %
(Intercept) -2.9796664 4.956695
x1          -0.6556513 1.500684
x2          -1.6182241 1.614749
x3          -0.4137027 1.845761
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> confint(object = modello, parm = c(2, 4), level = 0.99)

       0.5 %   99.5 %
x1 -1.365376 2.210409
x3 -1.157371 2.589429
```

Confint()

- **Package:** Rcmdr
- **Input:**

  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 $1-\alpha$

- **Description:** intervallo di confidenza per le stime OLS
- **Formula:**

$$\hat{\beta}_j \mp t_{1-\alpha\,/\,2,\,n-k}\, s_{\hat{\beta}_j} \quad \forall j = 1, 2, \ldots, k$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> Confint(object = modello, parm = c(1, 2, 3, 4), level = 0.95)
```

```
                 2.5 %   97.5 %
(Intercept) -2.9796664 4.956695
x1          -0.6556513 1.500684
x2          -1.6182241 1.614749
x3          -0.4137027 1.845761
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> Confint(object = modello, parm = c(2, 4), level = 0.99)
```

```
       0.5 %   99.5 %
x1 -1.365376 2.210409
x3 -1.157371 2.589429
```

coef()

- **Package:** stats
- **Input:**

  object modello di regressione lineare con $k - 1$ variabili esplicative ed n unità

- **Description:** stime OLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> coef(object = modello)
```

```
 (Intercept)           x1           x2           x3
 0.988514333  0.422516384 -0.001737381  0.716029046
```

coefficients()

- **Package:** stats
- **Input:**

  object modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** stime OLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> coefficients(object = modello)

 (Intercept)          x1           x2           x3
 0.988514333  0.422516384 -0.001737381  0.716029046
```

coeftest()

- **Package:** lmtest
- **Input:**

  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

- **Description:** stima puntuale, standard error, t-value, p-value
- **Formula:**

df = NULL

$$\hat{\beta}_j \qquad s_{\hat{\beta}_j} \qquad t_{\hat{\beta}_j} \qquad p\text{-value} = 2\,P(t_{n-k} \leq -\,|\,t_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1, 2, \ldots, k$$

df = Inf

$$\hat{\beta}_j \qquad s_{\hat{\beta}_j} \qquad z_{\hat{\beta}_j} \qquad p\text{-value} = 2\,\Phi\left(-\,|\,z_{\hat{\beta}_j}\,|\right) \qquad \forall\, j = 1, 2, \ldots, k$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> coeftest(x = modello, df = NULL)

t test of coefficients:

              Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.9885143  1.4292308  0.6916   0.5272
x1           0.4225164  0.3883267  1.0880   0.3377
x2          -0.0017374  0.5822146 -0.0030   0.9978
x3           0.7160290  0.4068987  1.7597   0.1533
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> coeftest(x = modello, df = Inf)

z test of coefficients:

              Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.9885143  1.4292308  0.6916  0.48916
x1           0.4225164  0.3883267  1.0880  0.27658
x2          -0.0017374  0.5822146 -0.0030  0.99762
x3           0.7160290  0.4068987  1.7597  0.07845 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- **Note:** Naturalmente vale che $t_{\hat{\beta}_j} = z_{\hat{\beta}_j} \quad \forall\, j = 1, 2, \ldots, k$.

boxcox()

- **Package:** MASS
- **Input:**

  `object` modello di regressione lineare con $k - 1$ variabili esplicative ed n unità

  `lambda` parametro di trasformazione λ

  `plotit = FALSE`

- **Description:** 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.

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> res <- boxcox(object = modello, lambda = 1.2, plotit = FALSE)
> res$x
```

```
[1] 1.2

> res$y

[1] -7.185995
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> res <- boxcox(object = modello, lambda = 4.1, plotit = FALSE)
> res$x

[1] 4.1

> res$y

[1] -9.591145
```

box.cox()

- **Package:** car
- **Input:**

  y vettore numerico positivo di dimensione n

  p parametro di trasformazione λ

- **Description:** variabile y trasformata secondo *Box–Cox*
- **Formula:**

$$t_\lambda(y) = \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{se } \lambda \neq 0 \\ \log(y) & \text{se } \lambda = 0 \end{cases}$$

- **Example 1:**

```
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> box.cox(y, p = 0.5)

[1] 0.4494897 3.0596443 4.1967734 3.9329588 3.9531504 3.5856960 3.8651513
[8] 3.8651513
```

- **Example 2:**

```
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> box.cox(y, p = 2)

[1]  0.6250 19.9800 45.5800 38.2200 38.7498 29.9200 36.4800 36.4800
```

box.cox.var()

- **Package:** car
- **Input:**

  y vettore numerico positivo di dimensione n

- **Description:** variabile y trasformata secondo *Box–Cox*
- **Formula:**

$$y_i \left(\log\left(y_i \,/\, \bar{y}_G\right) - 1\right) \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

$$\text{dove} \quad \bar{y}_G \,=\, \left(\prod_{i=1}^{n} y_i\right)^{1\,/\,n} \,=\, \exp\left(\frac{1}{n}\sum_{i=1}^{n}\log(y_i)\right)$$

- **Examples:**

```
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> box.cox.var(y)

[1] -3.748828 -6.709671 -6.172042 -6.423405 -6.406997 -6.634371 -6.475128
[8] -6.475128
```

bc()

- **Package:** car
- **Input:**

  y vettore numerico positivo di dimensione n

  p parametro di trasformazione λ

- **Description:** variabile y trasformata secondo *Box–Cox*
- **Formula:**

$$t_\lambda(y) \,=\, \begin{cases} \frac{y^\lambda - 1}{\lambda} & \text{se } \lambda \,\neq\, 0 \\ \log(y) & \text{se } \lambda \,=\, 0 \end{cases}$$

- **Example 1:**

```
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> bc(y, p = 0.5)

[1] 0.4494897 3.0596443 4.1967734 3.9329588 3.9531504 3.5856960 3.8651513
[8] 3.8651513
```

- **Example 2:**

```
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> bc(y, p = 2)

[1]  0.6250 19.9800 45.5800 38.2200 38.7498 29.9200 36.4800 36.4800
```

fitted()

- **Package:** stats
- **Input:**

  object modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{y}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> fitted(object = modello)

        1         2         3         4         5         6         7         8
 2.453638  5.964158  8.293288  8.102518  8.602437  7.139221  9.569117 10.035623
```

fitted.values()

- **Package:** stats
- **Input:**

  object modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{y}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> fitted.values(object = modello)

        1         2         3         4         5         6         7         8
 2.453638  5.964158  8.293288  8.102518  8.602437  7.139221  9.569117 10.035623
```

predict.lm()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` 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 $1-\alpha$

- **Description:** 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`

$$s^*$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> x0 <- c(1, 1.3, 2.1, 2.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat
```

```
[1] 3.181004
```

```
> new <- data.frame(x1 = 1.3, x2 = 2.1, x3 = 2.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> lower <- yhat - qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     X) %*% x0)
> upper <- yhat + qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     X) %*% x0)
> c(yhat, lower, upper)
```

```
[1] 3.181004 1.200204 5.161803

> res <- predict.lm(object = modello, newdata = new, se.fit = TRUE,
+     scale = s, df = Inf, interval = "confidence", level = 0.95)
> res$fit

       fit      lwr      upr
1 3.181004 1.200204 5.161803

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% X) %*%
+     x0))
> se.fit

[1] 1.010631

> res$se.fit

[1] 1.010631

> s

[1] 1.303508

> res$residual.scale

[1] 1.303508
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> x0 <- c(1, 1.3, 2.1, 2.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat

[1] 3.181004

> new <- data.frame(x1 = 1.3, x2 = 2.1, x3 = 2.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> lower <- yhat - qt(1 - 0.05/2, df = n - k) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% X) %*% x0)
> upper <- yhat + qt(1 - 0.05/2, df = n - k) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% X) %*% x0)
> c(yhat, lower, upper)

[1]  3.181004 -1.398453  7.760461

> res <- predict.lm(object = modello, newdata = new, se.fit = TRUE,
+     interval = "prediction", level = 0.95)
> res$fit

       fit       lwr      upr
1 3.181004 -1.398453 7.760461
```

```
> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% X) %*%
+     x0))
> se.fit

[1] 1.010631

> res$se.fit

[1] 1.010631

> s

[1] 1.303508

> res$residual.scale

[1] 1.303508
```

- **Note 1:** Per il calcolo dell'intervallo classico di confidenza o previsione impostare i parametri `df = n - k` e `scale = summary.lm(object = modello)$sigma`.
- **Note 2:** Per il calcolo dell'intervallo asintotico di confidenza o previsione impostare i parametri `df = Inf` e `scale = summary.lm(object = modello)$sigma`.

predict()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` 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 $1-\alpha$

- **Description:** 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

$$s^*$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> x0 <- c(1, 1.3, 2.1, 2.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat

[1] 3.181004

> new <- data.frame(x1 = 1.3, x2 = 2.1, x3 = 2.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> lower <- yhat - qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     X) %*% x0)
> upper <- yhat + qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     X) %*% x0)
> c(yhat, lower, upper)

[1] 3.181004 1.200204 5.161803

> res <- predict(object = modello, newdata = new, se.fit = TRUE,
+     scale = s, df = Inf, interval = "confidence", level = 0.95)
> res$fit

       fit      lwr      upr
1 3.181004 1.200204 5.161803

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% X) %*%
+     x0))
> se.fit

[1] 1.010631

> res$se.fit

[1] 1.010631

> s

[1] 1.303508

> res$residual.scale

[1] 1.303508
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> x0 <- c(1, 1.3, 2.1, 2.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat

[1] 3.181004

> new <- data.frame(x1 = 1.3, x2 = 2.1, x3 = 2.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> lower <- yhat - qt(1 - 0.05/2, df = n - k) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% X) %*% x0)
> upper <- yhat + qt(1 - 0.05/2, df = n - k) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% X) %*% x0)
> c(yhat, lower, upper)

[1]  3.181004 -1.398453  7.760461

> res <- predict(object = modello, newdata = new, se.fit = TRUE,
+     interval = "prediction", level = 0.95)
> res$fit

       fit       lwr      upr
1 3.181004 -1.398453 7.760461

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% X) %*%
+     x0))
> se.fit

[1] 1.010631

> res$se.fit

[1] 1.010631

> s

[1] 1.303508

> res$residual.scale

[1] 1.303508
```

- **Note 1:** Per il calcolo dell'intervallo classico di confidenza o previsione impostare i parametri `df = n - k` e `scale = summary.lm(object = modello)$sigma`.
- **Note 2:** Per il calcolo dell'intervallo asintotico di confidenza o previsione impostare i parametri `df = Inf` e `scale = summary.lm(object = modello)$sigma`.

linear.hypothesis()

- **Package:** car
- **Input:**

  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

- **Description:** 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)

$$P(F_{q,\,n-k} \geq Fvalue)$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> C <- matrix(data = c(1, 3, 5, 2.3, 2, 4, 1.1, 4.3), nrow = 2,
+     ncol = 4, byrow = TRUE)
> C

     [,1] [,2] [,3] [,4]
[1,]    1    3  5.0  2.3
[2,]    2    4  1.1  4.3
```

```
> b <- c(1.1, 2.3)
> b

[1] 1.1 2.3

> linear.hypothesis(model = modello, hypothesis.matrix = C, rhs = b)

Linear hypothesis test

Hypothesis:
(Intercept) + 3 x1 + 5 x2 + 2.3 x3 = 1.1
2 (Intercept) + 4 x1 + .1 x2 + 4.3 x3 = 2.3

Model 1: y ~ x1 + x2 + x3
Model 2: restricted model

  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1      4  6.7965
2      6 17.9679 -2  -11.1713 3.2874 0.1431

> res <- linear.hypothesis(model = modello, hypothesis.matrix = C,
+     rhs = b)
> q <- 2
> c(n - k, n - k + q)

[1] 4 6

> res$Res.Df

[1] 4 6

> X <- model.matrix(object = modello)
> RSS <- sum(residuals(object = modello)^2)
> beta <- coefficients(object = modello)
> CSS <- as.numeric(t(b - C %*% beta) %*% solve(C %*% solve(t(X) %*%
+     X) %*% t(C)) %*% (b - C %*% beta))
> c(RSS, RSS + CSS)

[1]  6.796529 17.967863

> res$RSS

[1]  6.796529 17.967863

> -q

[1] -2

> res$Df

[1] NA -2

> -CSS

[1] -11.17133

> res$"Sum of Sq"

[1]        NA -11.17133
```

```
> Fvalue <- (CSS/q)/(RSS/(n - k))
> Fvalue

[1] 3.287364

> res$F

[1]       NA 3.287364

> 1 - pf(Fvalue, df1 = q, df2 = n - k)

[1] 0.1430808

> res$"Pr(>F)"

[1]        NA 0.1430808
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> C <- matrix(data = c(1, 3, 5, 2.3, 2, 4, 1.1, 4.3, 12.3, 3.4,
+     4.5, 6.9), nrow = 3, ncol = 4, byrow = TRUE)
> C

     [,1] [,2] [,3] [,4]
[1,]  1.0  3.0  5.0  2.3
[2,]  2.0  4.0  1.1  4.3
[3,] 12.3  3.4  4.5  6.9

> b <- c(1.1, 2.3, 5.6)
> b

[1] 1.1 2.3 5.6

> linear.hypothesis(model = modello, hypothesis.matrix = C, rhs = b)

Linear hypothesis test

Hypothesis:
(Intercept) + 3 x1 + 5 x2 + 2.3 x3 = 1.1
2 (Intercept) + 4 x1 + .1 x2 + 4.3 x3 = 2.3
2.3 (Intercept) + 3.4 x1 + 4.5 x2 + 6.9 x3 = 5.6

Model 1: y ~ x1 + x2 + x3
Model 2: restricted model

  Res.Df     RSS Df Sum of Sq      F   Pr(>F)
1      4   6.797
2      7 109.041 -3  -102.244 20.058 0.007131 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> res <- linear.hypothesis(model = modello, hypothesis.matrix = C,
+     rhs = b)
> q <- 3
> c(n - k, n - k + q)
```

```
[1] 4 7

> res$Res.Df

[1] 4 7

> X <- model.matrix(object = modello)
> RSS <- sum(residuals(object = modello)^2)
> beta <- coefficients(object = modello)
> CSS <- as.numeric(t(b - C %*% beta) %*% solve(C %*% solve(t(X) %*%
+     X) %*% t(C)) %*% (b - C %*% beta))
> c(RSS, RSS + CSS)

[1]   6.796529 109.040699

> res$RSS

[1]   6.796529 109.040699

> -q

[1] -3

> res$Df

[1] NA -3

> -CSS

[1] -102.2442

> res$"Sum of Sq"

[1]        NA -102.2442

> Fvalue <- (CSS/q)/(RSS/(n - k))
> Fvalue

[1] 20.05811

> res$F

[1]       NA 20.05811

> 1 - pf(Fvalue, df1 = q, df2 = n - k)

[1] 0.007131315

> res$"Pr(>F)"

[1]          NA 0.007131315
```

lht()

- **Package:** car
- **Input:**

  `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

- **Description:** 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)`

$$P(F_{q,\,n-k} \geq Fvalue)$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> C <- matrix(data = c(1, 3, 5, 2.3, 2, 4, 1.1, 4.3), nrow = 2,
+     ncol = 4, byrow = TRUE)
> C
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    3  5.0  2.3
[2,]    2    4  1.1  4.3
```

```
> b <- c(1.1, 2.3)
> b

[1] 1.1 2.3

> lht(model = modello, hypothesis.matrix = C, rhs = b)

Linear hypothesis test

Hypothesis:
(Intercept) + 3 x1 + 5 x2 + 2.3 x3 = 1.1
2 (Intercept) + 4 x1 + .1 x2 + 4.3 x3 = 2.3

Model 1: y ~ x1 + x2 + x3
Model 2: restricted model

  Res.Df      RSS Df Sum of Sq      F Pr(>F)
1      4   6.7965
2      6  17.9679 -2  -11.1713 3.2874 0.1431

> res <- lht(model = modello, hypothesis.matrix = C, rhs = b)
> q <- 2
> c(n - k, n - k + q)

[1] 4 6

> res$Res.Df

[1] 4 6

> X <- model.matrix(object = modello)
> RSS <- sum(residuals(object = modello)^2)
> beta <- coefficients(object = modello)
> CSS <- as.numeric(t(b - C %*% beta) %*% solve(C %*% solve(t(X) %*%
+     X) %*% t(C)) %*% (b - C %*% beta))
> c(RSS, RSS + CSS)

[1]  6.796529 17.967863

> res$RSS

[1]  6.796529 17.967863

> -q

[1] -2

> res$Df

[1] NA -2

> -CSS

[1] -11.17133

> res$"Sum of Sq"

[1]        NA -11.17133
```

```
> Fvalue <- (CSS/q)/(RSS/(n - k))
> Fvalue

[1] 3.287364

> res$F

[1]       NA 3.287364

> 1 - pf(Fvalue, df1 = q, df2 = n - k)

[1] 0.1430808

> res$"Pr(>F)"

[1]        NA 0.1430808
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> C <- matrix(data = c(1, 3, 5, 2.3, 2, 4, 1.1, 4.3, 12.3, 3.4,
+     4.5, 6.9), nrow = 3, ncol = 4, byrow = TRUE)
> C

     [,1] [,2] [,3] [,4]
[1,]  1.0  3.0  5.0  2.3
[2,]  2.0  4.0  1.1  4.3
[3,] 12.3  3.4  4.5  6.9

> b <- c(1.1, 2.3, 5.6)
> b

[1] 1.1 2.3 5.6

> lht(model = modello, hypothesis.matrix = C, rhs = b)

Linear hypothesis test

Hypothesis:
(Intercept) + 3 x1 + 5 x2 + 2.3 x3 = 1.1
2 (Intercept) + 4 x1 + .1 x2 + 4.3 x3 = 2.3
2.3 (Intercept) + 3.4 x1 + 4.5 x2 + 6.9 x3 = 5.6

Model 1: y ~ x1 + x2 + x3
Model 2: restricted model

  Res.Df     RSS Df Sum of Sq      F   Pr(>F)
1      4   6.797
2      7 109.041 -3  -102.244 20.058 0.007131 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> res <- lht(model = modello, hypothesis.matrix = C, rhs = b)
> q <- 3
> c(n - k, n - k + q)
```

```
[1] 4 7

> res$Res.Df

[1] 4 7

> X <- model.matrix(object = modello)
> RSS <- sum(residuals(object = modello)^2)
> beta <- coefficients(object = modello)
> CSS <- as.numeric(t(b - C %*% beta) %*% solve(C %*% solve(t(X) %*%
+     X) %*% t(C)) %*% (b - C %*% beta))
> c(RSS, RSS + CSS)

[1]   6.796529 109.040699

> res$RSS

[1]   6.796529 109.040699

> -q

[1] -3

> res$Df

[1] NA -3

> -CSS

[1] -102.2442

> res$"Sum of Sq"

[1]        NA -102.2442

> Fvalue <- (CSS/q)/(RSS/(n - k))
> Fvalue

[1] 20.05811

> res$F

[1]       NA 20.05811

> 1 - pf(Fvalue, df1 = q, df2 = n - k)

[1] 0.007131315

> res$"Pr(>F)"

[1]          NA 0.007131315
```

lm.ridge()

- **Package:** MASS
- **Input:**

  `formula` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `lambda` valore del parametro λ
- **Description:** 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`

$$\lambda \qquad \frac{(y-\bar{y})^T\,(I_n - U\,D\,(D^2 + \lambda\, I_{k-1})^{-1}\, D\, U^T)^2\,(y - \bar{y})}{\left(n - \sum_{i=1}^{k-1} \frac{D_{i,\,i}^2}{\lambda + D_{i,\,i}^2}\right)^2}$$

  `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`

$$\frac{n\,(k-3)}{n-k}\;\frac{(y-\bar{y})^T\,(I_n - U\,U^T)\,(y-\bar{y})}{(y-\bar{y})^T\,U\,U^T\,(y-\bar{y})}$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> res <- lm.ridge(formula = modello, lambda = 1.2)
> res$coef

       x1        x2        x3
0.6830048 0.5524354 1.1242182

> res$scales
```

```
      x1       x2       x3
2.412986 2.352359 2.195831

> res$lambda

[1] 1.2

> res$ym

[1] 7.52

> res$xm

    x1     x2     x3
4.9000 6.0125 6.2450

> res$GCV

      1.2
0.2049004

> res$kHKB

[1] 0.483875

> res$kLW

[1] 0.3325936
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> res <- lm.ridge(formula = modello, lambda = 3.78)
> res$coef

       x1        x2        x3
0.5765168 0.6291156 0.8724114

> res$scales

      x1       x2       x3
2.412986 2.352359 2.195831

> res$lambda

[1] 3.78

> res$ym

[1] 7.52

> res$xm
```

```
    x1     x2     x3
4.9000 6.0125 6.2450
```

```
> res$GCV
```

```
    3.78
0.2013841
```

```
> res$kHKB
```

```
[1] 0.483875
```

```
> res$kLW
```

```
[1] 0.3325936
```

- **Note 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()`.
- **Note 2:** I parametri stimati sono $k-1$ e non k (modello senza intercetta).

cov2cor()

- **Package:** `stats`
- **Input:**

  `V` matrice di covarianza delle stime OLS di dimensione $k \times k$

- **Description:** converte la matrice di covarianza nella matrice di correlazione
- **Formula:**

$$r_{\hat{\beta}_i\,\hat{\beta}_j} \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> V <- vcov(object = modello)
> cov2cor(V)
```

```
            (Intercept)         x1          x2         x3
(Intercept)  1.00000000 -0.1860100  0.07158062 -0.4632900
x1          -0.18600997  1.0000000 -0.82213982  0.4883764
x2           0.07158062 -0.8221398  1.00000000 -0.8022181
x3          -0.46329002  0.4883764 -0.80221810  1.0000000
```

14.3 Adattamento

logLik()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** log-verosimiglianza normale
- **Formula:**

$$\hat{\ell}$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> logLik(object = modello)

'log Lik.' -10.69939 (df=5)
```

durbin.watson()

- **Package:** car
- **Input:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

  `dw` valore empirico della statistica *D–W*

- **Formula:**

  `dw`

$$\sum_{i=2}^{n} (e_i - e_{i-1})^2 \,/\, RSS$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> res <- durbin.watson(model = modello)
> res$dw

[1] 0.9255503
```

AIC()

- **Package:** stats
- **Input:**

  object modello di regressione lineare con $k - 1$ variabili esplicative ed n unità

- **Description:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell} + 2\,(k+1)$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> AIC(object = modello)

[1] 31.39878
```

BIC()

- **Package:** nlme
- **Input:**

  object modello di regressione lineare con $k - 1$ variabili esplicative ed n unità

- **Description:** indice *BIC*
- **Formula:**

$$-2\,\hat{\ell} + (k+1)\,\log(n)$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> BIC(object = modello)

[1] 31.79599
```

extractAIC()

- **Package:** stats
- **Input:**

  fit modello di regressione lineare con $k - 1$ variabili esplicative ed n unità

- **Description:** numero di parametri del modello ed indice *AIC* generalizzato
- **Formula:**

$$k \qquad n\,\log(RSS\,/\,n) + 2\,k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> extractAIC(fit = modello)

[1] 4.000000 6.695764
```

deviance()

- **Package:** stats
- **Input:**

  object modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** devianza residua
- **Formula:**

$$RSS$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> deviance(object = modello)

[1] 6.796529
```

PRESS()

- **Package:** MPV
- **Input:**

  x modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** PRESS
- **Formula:**

$$\sum_{i=1}^{n} e_i^2 \,/\, (1-h_i)^2$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> PRESS(x = modello)

[1] 35.00228
```

drop1()

- **Package:** stats
- **Input:**

  object modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  scale selezione indice AIC oppure Cp

  test = "F"
- **Description:** 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 = 0

$$n\,\log\left(RSS\,/\,n\right) + 2\,k,\; n\,\log\left(RSS_{-x_j}\,/\,n\right) + 2\,(k-1) \quad \forall\, j \,=\, 1,\,2,\,\ldots,\,k-1$$

  Cp

  scale = s^2

$$k,\; \frac{RSS_{-x_j}}{RSS\,/\,(n-k)} + 2\,(k-1) - n \quad \forall\, j \,=\, 1,\,2,\,\ldots,\,k-1$$

  F value

$$F_j \,=\, \frac{RSS_{-x_j} - RSS}{RSS\,/\,(n-k)} \quad \forall\, j \,=\, 1,\,2,\,\ldots,\,k-1$$

  Pr(F)

$$P(F_{1,\,n-k} \geq F_j) \quad \forall\, j \,=\, 1,\,2,\,\ldots,\,k-1$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> modello <- lm(formula = y ~ x1 + x2 + x3)
> drop1(object = modello, scale = 0, test = "F")
```

```
Single term deletions

Model:
y ~ x1 + x2 + x3
       Df Sum of Sq     RSS     AIC   F value  Pr(F)
<none>                6.7965  6.6958
x1      1    2.0115  8.8080  6.7698    1.1838 0.3377
x2      1 1.513e-05  6.7965  4.6958 8.905e-06 0.9978
x3      1    5.2616 12.0581  9.2824    3.0966 0.1533

> res <- drop1(object = modello, scale = 0, test = "F")
> res$Df

[1] NA  1  1  1

> res$"Sum of Sq"

[1]           NA 2.011499e+00 1.513044e-05 5.261577e+00

> res$RSS

[1]  6.796529  8.808029  6.796544 12.058107

> res$AIC

[1] 6.695764 6.769777 4.695782 9.282365

> res$"F value"

[1]           NA 1.183839e+00 8.904801e-06 3.096626e+00

> res$"Pr(F)"

[1]        NA 0.3377443 0.9977619 0.1532663
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> modello <- lm(formula = y ~ x1 + x2 + x3)
> s <- summary.lm(object = modello)$sigma
> s

[1] 1.303508

> drop1(object = modello, scale = s^2, test = "F")

Single term deletions

Model:
y ~ x1 + x2 + x3

scale:  1.699132

       Df Sum of Sq     RSS     Cp   F value  Pr(F)
<none>                6.7965 4.0000
x1      1    2.0115  8.8080 3.1838    1.1838 0.3377
x2      1 1.513e-05  6.7965 2.0000 8.905e-06 0.9978
x3      1    5.2616 12.0581 5.0966    3.0966 0.1533
```

```
> res <- drop1(object = modello, scale = s^2, test = "F")
> res$Df

[1] NA  1  1  1

> res$"Sum of Sq"

[1]           NA 2.011499e+00 1.513044e-05 5.261577e+00

> res$RSS

[1]  6.796529  8.808029  6.796544 12.058107

> res$Cp

[1] 4.000000 3.183839 2.000009 5.096626

> res$"F value"

[1]           NA 1.183839e+00 8.904801e-06 3.096626e+00

> res$"Pr(F)"

[1]        NA 0.3377443 0.9977619 0.1532663
```

add1()

- **Package:** `stats`
- **Input:**

  `object` modello nullo di regressione lineare

  `scope` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `scale` selezione indice AIC oppure Cp

  `test = "F"`
- **Description:** 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`

$$RSS_{nullo} - RSS_{x_j} \quad \forall\, j\,=\,1,\,2,\,\ldots,\,k-1$$

  dove RSS_{x_j} rappresenta la devianza residua del modello con la sola variabile esplicativa x_j.

`RSS`

$$RSS_{nullo},\, RSS_{x_j} \quad \forall\, j \,=\, 1,\, 2,\, \dots,\, k-1$$

`AIC`

`scale = 0`

$$n\,\log\left(RSS_{nullo}\,/\,n\right) + 2,\; n\,\log\left(RSS_{x_j}\,/\,n\right) + 4 \quad \forall\, j \,=\, 1,\, 2,\, \dots,\, k-1$$

`Cp`

`scale` = s^2

$$\frac{RSS_{nullo}}{RSS\,/\,(n-k)} + 2 - n,\; \frac{RSS_{x_j}}{RSS\,/\,(n-k)} + 4 - n \quad \forall\, j \,=\, 1,\, 2,\, \dots,\, k-1$$

`F value`

$$F_j \,=\, \frac{RSS_{nullo} - RSS_{x_j}}{RSS_{x_j}\,/\,(n-2)} \quad \forall\, j \,=\, 1,\, 2,\, \dots,\, k-1$$

`Pr(F)`

$$P(F_{1,\,n-2} \geq F_j) \quad \forall\, j \,=\, 1,\, 2,\, \dots,\, k-1$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> nullo <- lm(formula = y ~ 1)
> modello <- lm(formula = y ~ x1 + x2 + x3)
> add1(object = nullo, scope = modello, scale = 0, test = "F")

Single term additions

Model:
y ~ 1
       Df Sum of Sq    RSS    AIC F value    Pr(F)
<none>               47.666 16.278
x1      1    26.149 21.518 11.915  7.2914 0.035564 *
x2      1    35.492 12.175  7.359 17.4911 0.005799 **
x3      1    34.691 12.975  7.869 16.0418 0.007077 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> res <- add1(object = nullo, scope = modello, scale = 0, test = "F")
> res$Df

[1] NA  1  1  1

> res$"Sum of Sq"

[1]       NA 26.14878 35.49165 34.69113

> res$RSS

[1] 47.66640 21.51762 12.17475 12.97527

> res$AIC

[1] 16.278282 11.915446  7.359380  7.868828
```

```
> res$"F value"

[1]        NA  7.291356 17.491113 16.041811

> res$"Pr(F)"

[1]          NA 0.035564122 0.005799048 0.007076764
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> nullo <- lm(formula = y ~ 1)
> modello <- lm(formula = y ~ x1 + x2 + x3)
> s <- summary.lm(object = modello)$sigma
> s

[1] 1.303508

> add1(object = nullo, scope = modello, scale = s^2, test = "F")

Single term additions

Model:
y ~ 1

scale:  1.699132

       Df Sum of Sq    RSS      Cp F value    Pr(F)
<none>               47.666 22.0534
x1      1    26.149 21.518  8.6639  7.2914 0.035564 *
x2      1    35.492 12.175  3.1653 17.4911 0.005799 **
x3      1    34.691 12.975  3.6364 16.0418 0.007077 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> res <- add1(object = nullo, scope = modello, scale = s^2, test = "F")
> res$Df

[1] NA  1  1  1

> res$"Sum of Sq"

[1]       NA 26.14878 35.49165 34.69113

> res$RSS

[1] 47.66640 21.51762 12.17475 12.97527

> res$Cp

[1] 22.053378  8.663889  3.165274  3.636408

> res$"F value"

[1]        NA  7.291356 17.491113 16.041811

> res$"Pr(F)"

[1]          NA 0.035564122 0.005799048 0.007076764
```

leaps()

- **Package:** leaps
- **Input:**

  x matrice del modello priva della prima colonna (intercetta) di dimensione $n \times (h-1)$

  y variabile dipendente

  method = "r2" / "adjr2" / "Cp" indice R^2, R^2_{adj}, C_p

  nbest = 1

- **Description:** Best Subsets
- **Output:**

  which variabili selezionate

  size numero di parametri

  r2 / adjr2 / Cp indice R^2, R^2_{adj}, C_p

- **Formula:**

size

$$k_j \quad \forall\, j = 1, 2, \ldots, h-1$$

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	$k_j = j+1$	$\binom{h-1}{j}$
.	.	.
.	.	.
$h-1$	$k_{h-1} = h$	$\binom{h-1}{h-1}$

r2

method = "r2"

$$R^2_j \quad \forall\, j = 1, 2, \ldots, h-1$$

R^2_j rappresenta il massimo R^2 tra i $\binom{h-1}{j}$ modelli di regressione con j variabili esplicative oppure k_j parametri.

adjr2

method = "adjr2"

$$\begin{aligned} R^2_{adj\,j} &= 1 - \frac{RSS\,/\,(n-k_j)}{RSS_{nullo}\,/\,(n-1)} \\ &= \frac{1-k_j}{n-k_j} + \frac{n-1}{n-k_j}\,R^2_j \qquad \forall\, j = 1, 2, \ldots, h-1 \end{aligned}$$

$R^2_{adj\,j}$ rappresenta il massimo R^2_{adj} tra i $\binom{h-1}{j}$ modelli di regressione con j variabili esplicative oppure k_j parametri.

Cp

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.

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> modello <- lm(formula = y ~ x1 + x2 + x3)
> X <- model.matrix(object = modello)
> A <- X[, -1]
> leaps(x = A, y, method = "r2", nbest = 1)

$which
      1     2     3
1 FALSE  TRUE FALSE
2  TRUE FALSE  TRUE
3  TRUE  TRUE  TRUE

$label
[1] "(Intercept)" "1"          "2"          "3"

$size
[1] 2 3 4

$r2
[1] 0.7445843 0.8574144 0.8574147

> res <- leaps(x = A, y, method = "r2", nbest = 1)
> res$which

      1     2     3
1 FALSE  TRUE FALSE
2  TRUE FALSE  TRUE
3  TRUE  TRUE  TRUE

> res$size

[1] 2 3 4

> res$r2

[1] 0.7445843 0.8574144 0.8574147
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> modello <- lm(formula = y ~ x1 + x2 + x3)
> X <- model.matrix(object = modello)
> A <- X[, -1]
> leaps(x = A, y, method = "adjr2", nbest = 1)
```

```
$which
      1     2     3
1 FALSE  TRUE FALSE
2  TRUE FALSE  TRUE
3  TRUE  TRUE  TRUE

$label
[1] "(Intercept)" "1"           "2"           "3"

$size
[1] 2 3 4

$adjr2
[1] 0.7020150 0.8003801 0.7504757
```

```
> res <- leaps(x = A, y, method = "adjr2", nbest = 1)
> res$which
```

```
      1     2     3
1 FALSE  TRUE FALSE
2  TRUE FALSE  TRUE
3  TRUE  TRUE  TRUE
```

```
> res$size
```

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

```
> res$adjr2
```

```
[1] 0.7020150 0.8003801 0.7504757
```

- **Example 3:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> modello <- lm(formula = y ~ x1 + x2 + x3)
> X <- model.matrix(object = modello)
> A <- X[, -1]
> leaps(x = A, y, method = "Cp", nbest = 1)
```

```
$which
      1     2     3
1 FALSE  TRUE FALSE
2  TRUE FALSE  TRUE
3  TRUE  TRUE  TRUE

$label
[1] "(Intercept)" "1"           "2"           "3"

$size
[1] 2 3 4

$Cp
[1] 3.165274 2.000009 4.000000
```

```
> res <- leaps(x = A, y, method = "Cp", nbest = 1)
> res$which
```

```
      1     2     3
1 FALSE  TRUE FALSE
2  TRUE FALSE  TRUE
3  TRUE  TRUE  TRUE
```

```
> res$size
```

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

```
> res$Cp
```

```
[1] 3.165274 2.000009 4.000000
```

- **Note 1:** Tutti i modelli contengono l'intercetta.
- **Note 2:** $R^2_{adj\,j}$ è una trasformazione lineare crescente di $R^2_j \quad \forall\, j = 1, 2, \ldots, h-1$.
- **Note 3:** Cp_j è una trasformazione lineare decrescente di $R^2_j \quad \forall\, j = 1, 2, \ldots, h-1$.

bptest()

- **Package:** `lmtest`
- **Input:**

  `formula` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

  `studentize = TRUE / FALSE` metodo di *Koenker*
- **Description:** 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 = TRUE`

$$v_i = e_i^2 - RSS\,/\,n \quad \forall\, i = 1, 2, \ldots, n$$

$$c = n\,\frac{v^T\,H\,v}{v^T\,v}$$

  `studentize = FALSE`

$$v_i = n\,e_i^2\,/\,RSS - 1 \quad \forall\, i = 1, 2, \ldots, n$$

$$c = \frac{1}{2}\,v^T\,H\,v$$

  `parameter`

$$df = k-1$$

  `p.value`

$$P(\chi^2_{df} \geq c)$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> bptest(formula = modello, studentize = TRUE)

        studentized Breusch-Pagan test

data:  modello
BP = 3.2311, df = 3, p-value = 0.3574

> res <- bptest(formula = modello, studentize = TRUE)
> res$statistic

      BP
3.231074

> res$parameter

df
 3

> res$p.value

       BP
0.3573517
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> bptest(formula = modello, studentize = FALSE)

        Breusch-Pagan test

data:  modello
BP = 0.9978, df = 3, p-value = 0.8018

> res <- bptest(formula = modello, studentize = FALSE)
> res$statistic

       BP
0.9977698

> res$parameter

df
 3

> res$p.value

       BP
0.8017916
```

14.4 Diagnostica

ls.diag()

- **Package:** stats
- **Input:**

  `ls.out` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** 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`
  $$h_i \quad \forall\, i = 1, 2, \ldots, n$$

  `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`
  $$(X^T\,X)^{-1}$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> res <- ls.diag(ls.out = modello)
> res$std.dev
```

```
[1] 1.303508

> res$hat

[1] 0.7695906 0.4163361 0.3791092 0.3154744 0.7283511 0.5539241 0.4302463
[8] 0.4069682

> res$std.res

[1] -1.5241225  0.4376576  1.2722093  0.6467323  0.3791111  0.7589935 -0.9849613
[8] -1.4301703

> res$stud.res

[1] -2.0384846  0.3884371  1.4278921  0.5918863  0.3343822  0.7104546 -0.9800972
[8] -1.7718134

> res$cooks

[1] 1.93972080 0.03415783 0.24706215 0.04819074 0.09633983 0.17883712 0.18315058
[8] 0.35091186

> res$dfits

[1] -3.7255223  0.3280660  1.1157578  0.4018144  0.5475321  0.7916935 -0.8516950
[8] -1.4677742

> res$correlation

            (Intercept)         x1          x2         x3
(Intercept)  1.00000000 -0.1860100  0.07158062 -0.4632900
x1          -0.18600997  1.0000000 -0.82213982  0.4883764
x2           0.07158062 -0.8221398  1.00000000 -0.8022181
x3          -0.46329002  0.4883764 -0.80221810  1.0000000

> res$std.err

                 [,1]
(Intercept) 1.4292308
x1          0.3883267
x2          0.5822146
x3          0.4068987

> res$cov.scaled

            (Intercept)          x1          x2          x3
(Intercept)  2.04270054 -0.10323710  0.05956359 -0.26942727
x1          -0.10323710  0.15079759 -0.18587712  0.07716815
x2           0.05956359 -0.18587712  0.33897378 -0.19004733
x3          -0.26942727  0.07716815 -0.19004733  0.16556652

> res$cov.unscaled

            (Intercept)          x1          x2          x3
(Intercept)  1.20220217 -0.06075872  0.0350553 -0.15856757
x1          -0.06075872  0.08874976 -0.1093953  0.04541621
x2           0.03505530 -0.10939532  0.1994982 -0.11184964
x3          -0.15856757  0.04541621 -0.1118496  0.09744180
```

cooks.distance()

- **Package:** stats
- **Input:**

  model modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> cooks.distance(model = modello)

         1          2          3          4          5          6          7
1.93972080 0.03415783 0.24706215 0.04819074 0.09633983 0.17883712 0.18315058
         8
0.35091186
```

cookd()

- **Package:** car
- **Input:**

  model modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> cookd(model = modello)

         1          2          3          4          5          6          7
1.93972080 0.03415783 0.24706215 0.04819074 0.09633983 0.17883712 0.18315058
         8
0.35091186
```

rstandard()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> rstandard(model = modello)

         1          2          3          4          5          6          7
-1.5241225  0.4376576  1.2722093  0.6467323  0.3791111  0.7589935 -0.9849613
         8
-1.4301703
```

rstandard.lm()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> rstandard.lm(model = modello)

         1          2          3          4          5          6          7
-1.5241225  0.4376576  1.2722093  0.6467323  0.3791111  0.7589935 -0.9849613
         8
-1.4301703
```

stdres()

- **Package:** MASS
- **Input:**

  `object` modello di regressione lineare con $k - 1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> stdres(object = modello)

         1          2          3          4          5          6          7
-1.5241225  0.4376576  1.2722093  0.6467323  0.3791111  0.7589935 -0.9849613
         8
-1.4301703
```

rstudent()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con $k - 1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> rstudent(model = modello)

         1          2          3          4          5          6          7
-2.0384846  0.3884371  1.4278921  0.5918863  0.3343822  0.7104546 -0.9800972
         8
-1.7718134
```

rstudent.lm()

- **Package:** stats
- **Input:**

  model modello di regressione lineare con $k - 1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> rstudent.lm(model = modello)
```

```
         1          2          3          4          5          6          7
-2.0384846  0.3884371  1.4278921  0.5918863  0.3343822  0.7104546 -0.9800972
         8
-1.7718134
```

studres()

- **Package:** MASS
- **Input:**

  object modello di regressione lineare con $k - 1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> studres(object = modello)
```

```
         1          2          3          4          5          6          7
-2.0384846  0.3884371  1.4278921  0.5918863  0.3343822  0.7104546 -0.9800972
         8
-1.7718134
```

lmwork()

- **Package:** MASS
- **Input:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Description:** diagnostica di regressione
- **Output:**

  `stdedv` stima di σ

  `stdres` residui standard

  `studres` residui studentizzati
- **Formula:**

  `stdedv`
  $$s$$
  `stdres`
  $$rstandard_i \quad \forall i = 1, 2, \dots, n$$
  `studres`
  $$rstudent_i \quad \forall i = 1, 2, \dots, n$$
- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> lmwork(object = modello)

$stdedv
[1] 1.303508

$stdres
         1          2          3          4          5          6          7
-1.5241225  0.4376576  1.2722093  0.6467323  0.3791111  0.7589935 -0.9849613
         8
-1.4301703

$studres
         1          2          3          4          5          6          7
-2.0384846  0.3884371  1.4278921  0.5918863  0.3343822  0.7104546 -0.9800972
         8
-1.7718134

> res <- lmwork(object = modello)
> res$stdedv

[1] 1.303508

> res$stdres

         1          2          3          4          5          6          7
-1.5241225  0.4376576  1.2722093  0.6467323  0.3791111  0.7589935 -0.9849613
         8
-1.4301703

> res$studres
```

```
          1           2           3           4           5           6           7
-2.0384846  0.3884371  1.4278921  0.5918863  0.3343822  0.7104546 -0.9800972
          8
-1.7718134
```

dffits()

- **Package:** stats
- **Input:**

  model modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Description:** dffits
- **Formula:**

$$rstudent_i \sqrt{\frac{h_i}{1-h_i}} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> dffits(model = modello)

          1           2           3           4           5           6           7
-3.7255223  0.3280660  1.1157578  0.4018144  0.5475321  0.7916935 -0.8516950
          8
-1.4677742
```

covratio()

- **Package:** stats
- **Input:**

  model modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Description:** covratio
- **Formula:**

$$cr_i \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> covratio(model = modello)

          1           2           3           4           5           6           7
 0.4238374  4.4498753  0.6395729  2.9682483 10.0502975  3.8036903  1.8260516
          8
 0.3038647
```

lm.influence()

- **Package:** stats
- **Input:**

  model modello di regressione lineare con $k - 1$ variabili esplicative ed n unità

- **Description:** 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

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

  coefficients

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = e_i\,(1 - h_i)^{-1}\,(X^T\,X)^{-1}_j\,X_i^T \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

  sigma

$$s_{-i} \quad \forall\, i = 1, 2, \ldots, n$$

  wt.res

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> lm.influence(model = modello)

$hat
        1         2         3         4         5         6         7         8
0.7695906 0.4163361 0.3791092 0.3154744 0.7283511 0.5539241 0.4302463 0.4069682

$coefficients
  (Intercept)          x1          x2          x3
1 -3.95445343  0.12758388  0.01022818  0.44042192
2  0.21929134  0.01923025 -0.12292616  0.08309302
3 -0.15505077  0.14594807 -0.39064531  0.32853997
4  0.10864633 -0.01436987  0.12965355 -0.11055404
5  0.06456839  0.14591697 -0.04391330 -0.06357315
6  0.27248353 -0.28472521  0.38742501 -0.16358023
7  0.36758841  0.18614884 -0.28071294  0.03129723
8  0.76981755 -0.23622669  0.37474061 -0.34716366

$sigma
        1         2         3         4         5         6         7         8
0.9745992 1.4686808 1.1613865 1.4242946 1.4778725 1.3925645 1.3099769 1.0521638

$wt.res
         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227
```

```
> res <- lm.influence(model = modello)
> res$hat

        1         2         3         4         5         6         7         8
0.7695906 0.4163361 0.3791092 0.3154744 0.7283511 0.5539241 0.4302463 0.4069682

> res$coefficients

  (Intercept)          x1          x2          x3
1 -3.95445343  0.12758388  0.01022818  0.44042192
2  0.21929134  0.01923025 -0.12292616  0.08309302
3 -0.15505077  0.14594807 -0.39064531  0.32853997
4  0.10864633 -0.01436987  0.12965355 -0.11055404
5  0.06456839  0.14591697 -0.04391330 -0.06357315
6  0.27248353 -0.28472521  0.38742501 -0.16358023
7  0.36758841  0.18614884 -0.28071294  0.03129723
8  0.76981755 -0.23622669  0.37474061 -0.34716366

> res$sigma

        1         2         3         4         5         6         7         8
0.9745992 1.4686808 1.1613865 1.4242946 1.4778725 1.3925645 1.3099769 1.0521638

> res$wt.res

         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227
```

influence()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** 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`

  $$h_i \quad \forall\, i = 1, 2, \ldots, n$$

  `coefficients`

  $$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = e_i\,(1-h_i)^{-1}\,(X^T\,X)^{-1}_j\,X_i^T \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

  `sigma`

  $$s_{-i} \quad \forall\, i = 1, 2, \ldots, n$$

  `wt.res`

  $$e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> influence(model = modello)

$hat
        1         2         3         4         5         6         7         8
0.7695906 0.4163361 0.3791092 0.3154744 0.7283511 0.5539241 0.4302463 0.4069682

$coefficients
  (Intercept)          x1          x2          x3
1 -3.95445343  0.12758388  0.01022818  0.44042192
2  0.21929134  0.01923025 -0.12292616  0.08309302
3 -0.15505077  0.14594807 -0.39064531  0.32853997
4  0.10864633 -0.01436987  0.12965355 -0.11055404
5  0.06456839  0.14591697 -0.04391330 -0.06357315
6  0.27248353 -0.28472521  0.38742501 -0.16358023
7  0.36758841  0.18614884 -0.28071294  0.03129723
8  0.76981755 -0.23622669  0.37474061 -0.34716366

$sigma
        1         2         3         4         5         6         7         8
0.9745992 1.4686808 1.1613865 1.4242946 1.4778725 1.3925645 1.3099769 1.0521638

$wt.res
         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227

> res <- influence(model = modello)
> res$hat

        1         2         3         4         5         6         7         8
0.7695906 0.4163361 0.3791092 0.3154744 0.7283511 0.5539241 0.4302463 0.4069682

> res$coefficients

  (Intercept)          x1          x2          x3
1 -3.95445343  0.12758388  0.01022818  0.44042192
2  0.21929134  0.01923025 -0.12292616  0.08309302
3 -0.15505077  0.14594807 -0.39064531  0.32853997
4  0.10864633 -0.01436987  0.12965355 -0.11055404
5  0.06456839  0.14591697 -0.04391330 -0.06357315
6  0.27248353 -0.28472521  0.38742501 -0.16358023
7  0.36758841  0.18614884 -0.28071294  0.03129723
8  0.76981755 -0.23622669  0.37474061 -0.34716366

> res$sigma

        1         2         3         4         5         6         7         8
0.9745992 1.4686808 1.1613865 1.4242946 1.4778725 1.3925645 1.3099769 1.0521638

> res$wt.res

         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227
```

residuals()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** residui
- **Formula:**

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> residuals(object = modello)

         1          2          3          4          5          6          7 
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173 
         8 
-1.4356227 
```

residuals.lm()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** residui
- **Formula:**

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> residuals.lm(object = modello)

         1          2          3          4          5          6          7 
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173 
         8 
-1.4356227 
```

residuals.default()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** residui
- **Formula:**

$$e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> residuals.default(object = modello)

         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227
```

resid()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** residui
- **Formula:**

$$e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> resid(object = modello)

         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227
```

df.residual()

- **Package:** stats
- **Input:**

  object modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Description:** gradi di libertà della devianza residua
- **Formula:**

$$n - k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> df.residual(object = modello)

[1] 4
```

hatvalues()

- **Package:** stats
- **Input:**

  model modello di regressione lineare con $k-1$ variabili esplicative ed n unità
- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> hatvalues(model = modello)

        1         2         3         4         5         6         7         8
0.7695906 0.4163361 0.3791092 0.3154744 0.7283511 0.5539241 0.4302463 0.4069682
```

hat()

- **Package:** stats
- **Input:**

  x matrice del modello
- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> X <- model.matrix(object = modello)
> hat(x = X)

[1] 0.7695906 0.4163361 0.3791092 0.3154744 0.7283511 0.5539241 0.4302463
[8] 0.4069682
```

dfbeta()

- **Package:** stats
- **Input:**

  model modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** dfbeta
- **Formula:**

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} \;=\; e_i\,(1-h_i)^{-1}\,(X^T\,X)^{-1}_j\,X^T_i \quad \forall i \,=\, 1,\,2,\,\ldots,\,n \quad \forall j \,=\, 1,\,2,\,\ldots,\,k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> dfbeta(model = modello)

  (Intercept)          x1          x2          x3
1 -3.95445343  0.12758388  0.01022818  0.44042192
2  0.21929134  0.01923025 -0.12292616  0.08309302
3 -0.15505077  0.14594807 -0.39064531  0.32853997
4  0.10864633 -0.01436987  0.12965355 -0.11055404
5  0.06456839  0.14591697 -0.04391330 -0.06357315
6  0.27248353 -0.28472521  0.38742501 -0.16358023
7  0.36758841  0.18614884 -0.28071294  0.03129723
8  0.76981755 -0.23622669  0.37474061 -0.34716366
```

dfbetas()

- **Package:** stats
- **Input:**

  model modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** dfbetas
- **Formula:**

$$\frac{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}{s_{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}} \;=\; \frac{e_i\,(1-h_i)^{-1}\,(X^T\,X)^{-1}_j\,X^T_i}{s_{-i}\,\sqrt{(X^T\,X)^{-1}_{j,\,j}}} \quad \forall i \,=\, 1,\,2,\,\ldots,\,n \quad \forall j \,=\, 1,\,2,\,\ldots,\,k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> dfbetas(model = modello)

  (Intercept)          x1          x2          x3
1 -3.70059595  0.43942641  0.02349647  1.44767218
2  0.13617748  0.04395152 -0.18739044  0.18124433
3 -0.12176106  0.42183052 -0.75307182  0.90623075
4  0.06957072 -0.03386642  0.20380513 -0.24865783
5  0.03984687  0.33142498 -0.06652573 -0.13780473
6  0.17845806 -0.68632053  0.62287782 -0.37630746
7  0.25592307  0.47699422 -0.47976587  0.07653668
8  0.66729165 -0.75363662  0.79740312 -1.05700791
```

vif()

- **Package:** car
- **Input:**

  `mod` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** 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.

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> vif(mod = modello)

      x1       x2       x3
4.133964 8.831535 3.758662
```

outlier.test()

- **Package:** car
- **Input:**

  `model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** test sugli *outliers*
- **Output:**

`test` massimo residuo studentizzato assoluto, gradi di libertà, p-value

- **Formula:**

`test`

$$t = \max_i(\,|\,rstudent_i\,|) \quad n-k-1 \quad p\text{-value} = 2\,P(\,t_{n-k-1} \leq -|\,t\,|) \qquad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> outlier.test(model = modello)

max|rstudent| = 2.038485, degrees of freedom = 3,
unadjusted p = 0.1342423, Bonferroni p > 1

Observation: 1

> res <- outlier.test(model = modello)
> res$test

max|rstudent|            df  unadjusted p  Bonferroni p
    2.0384846     3.0000000     0.1342423            NA
```

influence.measures()

- **Package:** stats
- **Input:**

`model` modello di regressione lineare con $k-1$ variabili esplicative ed n unità

- **Description:** dfbetas, dffits, covratio, distanza di *Cook*, valori di leva
- **Output:**

`infmat` misure di influenza di dimensione $n \times (k+4)$

`is.inf` matrice di influenza con valori logici di dimensione $n \times (k+4)$

- **Formula:**

`infmat`

$$DFBETAS_{ij} = \frac{e_i\,(1-h_i)^{-1}\,(X^T\,X)_j^{-1}\,X_i^T}{s_{-i}\,\sqrt{(X^T\,X)_{j,\,j}^{-1}}} \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

$$DFFITS_i = rstudent_i\,\sqrt{\frac{h_i}{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$$

$$COVRATIO_i = (1-h_i)^{-1}\,\left(1 + \frac{rstudent_i^2 - 1}{n-k}\right)^{-\,k} \quad \forall i = 1, 2, \ldots, n$$

$$COOKD_i = \frac{h_i\,rstandard_i^2}{k\,(1-h_i)} \quad \forall\, i = 1, 2, \ldots, n$$

$$HAT_i = h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
```

```
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3)
> res <- influence.measures(model = modello)
> res

Influence measures of
         lm(formula = y ~ x1 + x2 + x3) :

   dfb.1_  dfb.x1  dfb.x2  dfb.x3  dffit  cov.r cook.d   hat inf
1 -3.7006  0.4394  0.0235  1.4477 -3.726  0.424 1.9397 0.770   *
2  0.1362  0.0440 -0.1874  0.1812  0.328  4.450 0.0342 0.416   *
3 -0.1218  0.4218 -0.7531  0.9062  1.116  0.640 0.2471 0.379
4  0.0696 -0.0339  0.2038 -0.2487  0.402  2.968 0.0482 0.315
5  0.0398  0.3314 -0.0665 -0.1378  0.548 10.050 0.0963 0.728   *
6  0.1785 -0.6863  0.6229 -0.3763  0.792  3.804 0.1788 0.554
7  0.2559  0.4770 -0.4798  0.0765 -0.852  1.826 0.1832 0.430
8  0.6673 -0.7536  0.7974 -1.0570 -1.468  0.304 0.3509 0.407   *

> res$infmat

       dfb.1_      dfb.x1      dfb.x2      dfb.x3      dffit      cov.r
1 -3.70059595  0.43942641  0.02349647  1.44767218 -3.7255223  0.4238374
2  0.13617748  0.04395152 -0.18739044  0.18124433  0.3280660  4.4498753
3 -0.12176106  0.42183052 -0.75307182  0.90623075  1.1157578  0.6395729
4  0.06957072 -0.03386642  0.20380513 -0.24865783  0.4018144  2.9682483
5  0.03984687  0.33142498 -0.06652573 -0.13780473  0.5475321 10.0502975
6  0.17845806 -0.68632053  0.62287782 -0.37630746  0.7916935  3.8036903
7  0.25592307  0.47699422 -0.47976587  0.07653668 -0.8516950  1.8260516
8  0.66729165 -0.75363662  0.79740312 -1.05700791 -1.4677742  0.3038647
      cook.d       hat
1 1.93972080 0.7695906
2 0.03415783 0.4163361
3 0.24706215 0.3791092
4 0.04819074 0.3154744
5 0.09633983 0.7283511
6 0.17883712 0.5539241
7 0.18315058 0.4302463
8 0.35091186 0.4069682

> res$is.inf

  dfb.1_ dfb.x1 dfb.x2 dfb.x3 dffit cov.r cook.d   hat
1   TRUE  FALSE  FALSE   TRUE  TRUE FALSE   TRUE FALSE
2  FALSE  FALSE  FALSE  FALSE FALSE  TRUE  FALSE FALSE
3  FALSE  FALSE  FALSE  FALSE FALSE FALSE  FALSE FALSE
4  FALSE  FALSE  FALSE  FALSE FALSE FALSE  FALSE FALSE
5  FALSE  FALSE  FALSE  FALSE FALSE  TRUE  FALSE FALSE
6  FALSE  FALSE  FALSE  FALSE FALSE FALSE  FALSE FALSE
7  FALSE  FALSE  FALSE  FALSE FALSE FALSE  FALSE FALSE
8  FALSE  FALSE  FALSE   TRUE FALSE FALSE  FALSE FALSE
```

- **Note 1:** Il caso i-esimo è influente se $|DFBETAS_{ij}| > 1 \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$
- **Note 2:** Il caso i-esimo è influente se $|DFFITS_i| > 3\sqrt{k \,/\, (n-k)} \quad \forall i = 1, 2, \ldots, n$
- **Note 3:** Il caso i-esimo è influente se $|1 - COVRATIO_i| > 3\,k \,/\, (n-k) \quad \forall i = 1, 2, \ldots, n$
- **Note 4:** Il caso i-esimo è influente se $P\,(F_{k,n-k} \geq COOKD_i) > 0.5 \quad \forall i = 1, 2, \ldots, n$
- **Note 5:** Il caso i-esimo è influente se $HAT_i > 3\,k \,/\, n \quad \forall i = 1, 2, \ldots, n$
- **Note 6:** I casi influenti rispetto ad almeno una tra queste misure sono marcati con un asterisco. Corrispondentemente la stessa riga della matrice `is.inf` riporterà almeno un simbolo `TRUE`.

Capitolo 15

Regressione lineare semplice pesata

15.1 Simbologia

$$y_i = \beta_1 + \beta_2\, x_i + \varepsilon_i \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n \qquad \varepsilon \sim N(0, \sigma^2\, W)$$

- 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 : $X_i \,=\, (1,\, x_i) \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- vettore numerico positivo dei pesi WLS: $w \,=\, (w_1,\, w_2,\, \ldots,\, w_n)$
- matrice diagonale definita positiva di dimensione $n \times n$: $W \,=\, \text{diag}(w_1^{-1},\, w_2^{-1},\, \ldots,\, w_n^{-1})$
- matrice di proiezione di dimensione $n \times n$: $H \,=\, X\,(X^T\, W^{-1}\, X)^{-1}\, X^T\, W^{-1}$
- matrice identità di dimensione $n \times n$: I_n
- devianza residua: $RSS \,=\, \sum_{i=1}^{n} w_i\, e_i^2 \,=\, y^T\, W^{-1}\, e \,=\, y^T\, W^{-1}\, (I_n - H)\, y$
- 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: $s_{-i}^2 \,=\, s^2\, \left(1 + \frac{1 - rstandard_i^2}{n-3}\right) \,=\, s^2\, \left(1 + \frac{rstudent_i^2 - 1}{n-2}\right)^{-1} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- codevianza pesata tra x ed y: $ss_{xy} \,=\, \sum_{i=1}^{n} w_i\, (x_i - \bar{x}_W)\, (y_i - \bar{y}_W)$
- devianza pesata di x: $ss_x \,=\, \sum_{i=1}^{n} w_i\, (x_i - \bar{x}_W)^2$
- devianza pesata di y: $ss_y \,=\, \sum_{i=1}^{n} w_i\, (y_i - \bar{y}_W)^2$
- stime WLS: $\hat{\beta} \,=\, (X^T\, W^{-1}\, X)^{-1}\, X^T\, W^{-1}\, y$
- stima WLS intercetta: $\hat{\beta}_1 \,=\, \bar{y}_W - \bar{x}_W\, ss_{xy}\,/\,ss_x$
- stima WLS coefficiente angolare: $\hat{\beta}_2 \,=\, ss_{xy}\,/\,ss_x$
- standard error delle stime WLS: $s_{\hat{\beta}} \,=\, s\, \sqrt{\text{diag}((X^T\, W^{-1}\, X)^{-1})}$
- standard error della stima WLS intercetta: $s_{\hat{\beta}_1} \,=\, s\, \sqrt{\sum_{i=1}^{n} w_i\, x_i^2\,/\,(ss_x\, \sum_{i=1}^{n} w_i)}$
- standard error della stima WLS coefficiente angolare: $s_{\hat{\beta}_2} \,=\, s\,/\,\sqrt{ss_x}$
- covarianza tra le stime WLS: $s_{\hat{\beta}_1\, \hat{\beta}_2} \,=\, -\bar{x}_W\, s^2\,/\,ss_x$
- t-values delle stime WLS: $t_{\hat{\beta}} \,=\, \hat{\beta}\,/\,s_{\hat{\beta}}$
- residui: $e \,=\, (I_n - H)\, y$
- residui pesati: $\sqrt{w_i}\, e_i \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$

- residui standard: $rstandard_i = \frac{e_i}{s\sqrt{(1-h_i)\,/\,w_i}} \quad \forall\, i = 1, 2, \ldots, n$
- residui studentizzati: $rstudent_i = \frac{e_i}{s_{-i}\sqrt{(1-h_i)\,/\,w_i}} = rstandard_i \sqrt{\frac{n-3}{n-2-rstandard_i^2}} \quad \forall\, i = 1, 2, \ldots, n$
- valori adattati: $\hat{y} = H\, y$
- valori di leva: $h_i = H_{i,i} \quad \forall\, i = 1, 2, \ldots, n$
- stime WLS tolta la i-esima unità: $\hat{\beta}_{(-i)} \quad \forall\, i = 1, 2, \ldots, n$
- correlazione delle stime WLS: $r_{\hat{\beta}_i\,\hat{\beta}_j} = \frac{s^2\,(X^T\,W^{-1}\,X)^{-1}_{(i,\,j)}}{s_{\hat{\beta}_i}\, s_{\hat{\beta}_j}} \quad \forall\, i, j = 1, 2$
- 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: $R^2 = 1 - RSS\,/\,RSS_{nullo} = 1 - (1 - R^2_{adj})\,(n-2)\,/\,(n-1) = r^2_{xy}$
- indice di determinazione aggiustato: $R^2_{adj} = 1 - \frac{RSS\,/\,(n-2)}{RSS_{nullo}\,/\,(n-1)} = 1 - \left(1 - R^2\right)\,(n-1)\,/\,(n-2)$
- valore noto dei regressori per la previsione: x_0
- log-verosimiglianza normale: $\hat{\ell} = -n\left(\log(2\,\pi) + \log\left(RSS\,/\,n\right) + 1 - \sum_{i=1}^n \log(w_i)\,/\,n\right)\,/\,2$
- distanza di *Cook*: $cd_i = \frac{h_i\, rstandard_i^2}{2\,(1-h_i)} = \frac{e_i^2}{2\,s^2}\,\frac{h_i}{(1-h_i)^2} \quad \forall\, i = 1, 2, \ldots, n$
- covratio: $cr_i = (1-h_i)^{-1}\left(1 + \frac{rstudent_i^2 - 1}{n-2}\right)^{-2} = (1-h_i)^{-1}\left(\frac{s_{-i}}{s}\right)^4 \quad \forall i = 1, 2, \ldots, n$

15.2 Stima

lm()

- **Package:** stats
- **Input:**

  `formula` modello di regressione lineare pesata con una variabile esplicativa ed n unità

  `weights` pesi

  `x = TRUE` matrice del modello

  `y = TRUE` variabile dipendente
- **Description:** analisi di regressione lineare pesata
- **Output:**

  `coefficients` stime WLS

  `residuals` residui

  `fitted.values` valori adattati

  `weights` pesi

  `rank` rango della matrice del modello

  `df.residual` gradi di libertà della devianza residua

  `x` matrice del modello

  `y` variabile dipendente
- **Formula:**

  `coefficients`

  $$\hat{\beta}_j \quad \forall\, j = 1, 2$$

  `residuals`

  $$e_i \quad \forall\, i = 1, 2, \ldots, n$$

  `fitted.values`

  $$\hat{y}_i \quad \forall\, i = 1, 2, \ldots, n$$

weights

$$w_i \quad \forall\, i = 1, 2, \ldots, n$$

rank

$$2$$

df.residual

$$n - 2$$

x

$$X$$

y

$$y$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n), x = TRUE,
+     y = TRUE)
> modello$coefficients

(Intercept)           x
  3.8486818   0.7492486

> modello$residuals

          1           2           3           4           5           6
-3.17285530  0.82804637  2.37969944 -0.06864749 -1.65699442  1.40387291
          7           8
 0.55552598 -0.26864749

> modello$fitted.values

        1         2         3         4         5         6         7         8
 4.672855  5.571954  7.220301  8.868647 10.516994  6.396127  8.044474  8.868647

> modello$weights

[1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125

> modello$rank

[1] 2

> modello$df.residual

[1] 6

> modello$x

  (Intercept)   x
1           1 1.1
2           1 2.3
3           1 4.5
4           1 6.7
5           1 8.9
6           1 3.4
7           1 5.6
8           1 6.7
attr(,"assign")
[1] 0 1
```

```
> modello$y

   1    2    3    4    5    6    7    8
1.50 6.40 9.60 8.80 8.86 7.80 8.60 8.60
```

- **Note 1:** Il modello nullo si ottiene attraverso con `lm(formula = y ~ 1,weights = w)`.
- **Note 2:** L'istruzione `lm(formula = y ~ x,weights = w)` è equivalente a `lm(formula = y ~ X - 1,weight`
- **Note 3:** L'istruzione `lm(formula = y ~ x,weights = w)` è equivalente a `lm(formula = y ~ 1 + x,weight`

summary.lm()

- **Package:** `stats`
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

  `correlation = TRUE` correlazione delle stime WLS
- **Description:** analisi di regressione lineare pesata
- **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 WLS non scalata per σ^2

  `correlation` matrice di correlazione delle stime WLS
- **Formula:**

  `residuals`

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

  `coefficients`

$$\hat{\beta}_j \qquad s_{\hat{\beta}_j} \qquad t_{\hat{\beta}_j} \qquad p\text{-value} = 2\,P(t_{n-2} \leq -\,|\,t_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1, 2$$

  `sigma`

$$s$$

  `r.squared`

$$R^2$$

  `adj.r.squared`

$$R^2_{adj}$$

  `fstatistic`

$$F value = \frac{RSS_{nullo} - RSS}{RSS\,/\,(n-2)} = t^2_{\hat{\beta}_2} \qquad 1 \qquad n-2$$

  `cov.unscaled`

$$(X^T\, W^{-1}\, X)^{-1}$$

  `correlation`

$$r_{\hat{\beta}_i\,\hat{\beta}_j} \quad \forall\, i, j = 1, 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> res <- summary.lm(object = modello, correlation = TRUE)
> res$residuals
```

```
          1           2           3           4           5           6
-1.12177375  0.29275860  0.84135081 -0.02427055 -0.58583599  0.49634403
          7           8
 0.19640809 -0.09498123
```

```
> res$coefficients
```

```
             Estimate Std. Error  t value   Pr(>|t|)
(Intercept) 3.8486818  1.5155372 2.539484 0.04411163
x           0.7492486  0.2774737 2.700251 0.03556412
```

```
> res$sigma
```

```
[1] 0.66954
```

```
> res$r.squared
```

```
[1] 0.5485788
```

```
> res$adj.r.squared
```

```
[1] 0.4733419
```

```
> res$fstatistic
```

```
   value    numdf    dendf
7.291356 1.000000 6.000000
```

```
> res$cov.unscaled
```

```
            (Intercept)          x
(Intercept)   5.1236582 -0.8415629
x            -0.8415629  0.1717475
```

```
> res$correlation
```

```
            (Intercept)          x
(Intercept)   1.0000000 -0.8971215
x            -0.8971215  1.0000000
```

vcov()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** matrice di covarianza delle stime WLS
- **Formula:**

$$s^2 \, (X^T \, W^{-1} \, X)^{-1}$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> vcov(object = modello)

            (Intercept)           x
(Intercept)   2.2968531 -0.37725904
x            -0.3772590  0.07699164
```

lm.wfit()

- **Package:** stats
- **Input:**

  `x` matrice del modello

  `y` variabile dipendente

  `w` pesi

- **Description:** analisi di regressione lineare pesata
- **Output:**

  `coefficients` stime WLS

  `residuals` residui

  `fitted.values` valori adattati

  `weights` pesi

  `rank` rango della matrice del modello

  `df.residual` gradi di libertà della devianza residua

- **Formula:**

  `coefficients`

$$\hat{\beta}_j \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k$$

  `residuals`

$$e_i \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

  `fitted.values`

$$\hat{y}_i \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

  `weights`

$$w_i \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

  `rank`

$$k$$

  `df.residual`

$$n - k$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> X <- model.matrix(object = modello)
> res <- lm.wfit(x = X, y, w = rep(1/n, n))
> res$coefficients

(Intercept)           x
  3.8486818   0.7492486

> res$residuals

[1] -3.17285530  0.82804637  2.37969944 -0.06864749 -1.65699442  1.40387291
[7]  0.55552598 -0.26864749

> res$fitted.values

[1]  4.672855  5.571954  7.220301  8.868647 10.516994  6.396127  8.044474
[8]  8.868647

> res$weights

[1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125

> res$rank

[1] 2

> res$df.residual

[1] 6
```

lsfit()

- **Package:** stats
- **Input:**

  x matrice del modello

  y variabile dipendente

  wt pesi

  intercept = FALSE

- **Description:** analisi di regressione lineare pesata
- **Output:**

  coefficients stime WLS

  residuals residui

  wt pesi

- **Formula:**

  coefficients

$$\hat{\beta}_j \quad \forall\, j = 1, 2$$

residuals

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

wt

$$w_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> X <- model.matrix(object = modello)
> res <- lsfit(x = X, y, wt = rep(1/n, n), intercept = FALSE)
> res$coefficients

(Intercept)           x
  3.8486818   0.7492486


> res$residuals


[1] -3.17285530  0.82804637  2.37969944 -0.06864749 -1.65699442  1.40387291
[7]  0.55552598 -0.26864749


> res$wt


[1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125
```

confint()

- **Package:** stats
- **Input:**

  object modello di regressione lineare pesata con una variabile esplicativa ed n unità

  parm parametri del modello su cui calcolare l'intervallo di confidenza

  level livello di confidenza $1 - \alpha$

- **Description:** intervallo di confidenza per le stime WLS
- **Formula:**

$$\hat{\beta}_j \mp t_{1-\alpha\,/\,2,\, n-2}\, s_{\hat{\beta}_j} \quad \forall\, j = 1, 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> confint(object = modello, parm = c(1, 2), level = 0.95)

                 2.5 %   97.5 %
(Intercept) 0.14029581 7.557068
x           0.07029498 1.428202
```

coef()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** stime WLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j \,=\, 1,\, 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> coef(object = modello)

(Intercept)           x
  3.8486818   0.7492486
```

fitted()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{y}_i \quad \forall\, i \,=\, 1,\, 2,\, \dots,\, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> fitted(object = modello)

        1         2         3         4          5         6         7         8
 4.672855  5.571954  7.220301  8.868647 10.516994  6.396127  8.044474  8.868647
```

predict.lm()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` 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 $1 - \alpha$

- **Description:** 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"`

  $$x_0^T\,\hat{\beta} \qquad x_0^T\,\hat{\beta} \mp t_{1-\alpha\,/\,2,\,df}\; s^*\,\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

  `interval = "prediction"`

  $$x_0^T\,\hat{\beta} \qquad x_0^T\,\hat{\beta} \mp t_{1-\alpha\,/\,2,\,df}\; s^*\,\sqrt{1 + x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

  `se.fit`

  $$s^*\,\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

  `df`

  $$df \;=\; n-2$$

  `residual.scale`

  $$s^*$$
- **Example 1:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> x0 <- c(1, 1.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat

[1] 4.822705

> new <- data.frame(x = 1.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> W <- diag(1/rep(1/n, n))
> lower <- yhat - qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     solve(W) %*% X) %*% x0)
> upper <- yhat + qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     solve(W) %*% X) %*% x0)
> c(yhat, lower, upper)

[1] 4.822705 2.465776 7.179634

> res <- predict.lm(object = modello, newdata = new, se.fit = TRUE,
+     scale = s, df = Inf, interval = "confidence", level = 0.95)
> res$fit

       fit      lwr      upr
1 4.822705 2.465776 7.179634

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% solve(W) %*%
+     X) %*% x0))
> se.fit
```

```
[1] 1.202537

> res$se.fit

[1] 1.202537

> s

[1] 0.66954

> res$residual.scale

[1] 0.66954
```

- **Example 2:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> x0 <- c(1, 1.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat

[1] 4.822705

> new <- data.frame(x = 1.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> W <- diag(1/rep(1/n, n))
> lower <- yhat - qt(1 - 0.05/2, df = n - 2) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% solve(W) %*% X) %*% x0)
> upper <- yhat + qt(1 - 0.05/2, df = n - 2) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% solve(W) %*% X) %*% x0)
> c(yhat, lower, upper)

[1] 4.822705 1.454862 8.190548

> res <- predict.lm(object = modello, newdata = new, se.fit = TRUE,
+     interval = "prediction", level = 0.95)
> res$fit

       fit      lwr      upr
1 4.822705 1.454862 8.190548

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% solve(W) %*%
+     X) %*% x0))
> se.fit

[1] 1.202537

> res$se.fit

[1] 1.202537

> s

[1] 0.66954

> res$residual.scale
```

```
[1] 0.66954
```

- **Note 1:** Per il calcolo dell'intervallo classico di confidenza o previsione impostare i parametri `df = n - 2` e `scale = summary.lm(object = modello)$sigma`.
- **Note 2:** Per il calcolo dell'intervallo asintotico di confidenza o previsione impostare i parametri `df = Inf` e `scale = summary.lm(object = modello)$sigma`.

predict()

- **Package:** `stats`
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` 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 $1 - \alpha$
- **Description:** 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"`

$$x_0^T \hat{\beta} \qquad x_0^T \hat{\beta} \mp t_{1-\alpha\,/\,2,\, df}\; s^* \sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

  `interval = "prediction"`

$$x_0^T \hat{\beta} \qquad x_0^T \hat{\beta} \mp t_{1-\alpha\,/\,2,\, df}\; s^* \sqrt{1 + x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

  `se.fit`

$$s^* \sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

  `df`

$$df = n - 2$$

  `residual.scale`

$$s^*$$

- **Example 1:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> x0 <- c(1, 1.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat
```

```
[1] 4.822705
```

```
> new <- data.frame(x = 1.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> W <- diag(1/rep(1/n, n))
> lower <- yhat - qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     solve(W) %*% X) %*% x0)
> upper <- yhat + qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     solve(W) %*% X) %*% x0)
> c(yhat, lower, upper)

[1] 4.822705 2.465776 7.179634

> res <- predict(object = modello, newdata = new, se.fit = TRUE,
+     scale = s, df = Inf, interval = "confidence", level = 0.95)
> res$fit

       fit      lwr      upr
1 4.822705 2.465776 7.179634

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% solve(W) %*%
+     X) %*% x0))
> se.fit

[1] 1.202537

> res$se.fit

[1] 1.202537

> s

[1] 0.66954

> res$residual.scale

[1] 0.66954
```

- **Example 2:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> x0 <- c(1, 1.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat

[1] 4.822705

> new <- data.frame(x = 1.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> W <- diag(1/rep(1/n, n))
> lower <- yhat - qt(1 - 0.05/2, df = n - 2) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% solve(W) %*% X) %*% x0)
> upper <- yhat + qt(1 - 0.05/2, df = n - 2) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% solve(W) %*% X) %*% x0)
> c(yhat, lower, upper)

[1] 4.822705 1.454862 8.190548
```

```
> res <- predict(object = modello, newdata = new, se.fit = TRUE,
+      interval = "prediction", level = 0.95)
> res$fit

        fit      lwr      upr
1 4.822705 1.454862 8.190548

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% solve(W) %*%
+      X) %*% x0))
> se.fit

[1] 1.202537

> res$se.fit

[1] 1.202537

> s

[1] 0.66954

> res$residual.scale

[1] 0.66954
```

- **Note 1:** Per il calcolo dell'intervallo classico di confidenza o previsione impostare i parametri `df = n - 2` e `scale = summary.lm(object = modello)$sigma`.
- **Note 2:** Per il calcolo dell'intervallo asintotico di confidenza o previsione impostare i parametri `df = Inf` e `scale = summary.lm(object = modello)$sigma`.

cov2cor()

- **Package:** `stats`
- **Input:**

  `V` matrice di covarianza delle stime WLS di dimensione 2×2

- **Description:** converte la matrice di covarianza nella matrice di correlazione
- **Formula:**

$$r_{\hat{\beta}_i \, \hat{\beta}_j} \quad \forall \, i, j \, = \, 1, \, 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> V <- vcov(object = modello)
> cov2cor(V)

            (Intercept)          x
(Intercept)   1.0000000 -0.8971215
x            -0.8971215  1.0000000
```

15.3 Adattamento

logLik()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** log-verosimiglianza normale
- **Formula:**

$$\hat{\ell}$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> logLik(object = modello)

'log Lik.' -15.30923 (df=3)
```

durbin.watson()

- **Package:** car
- **Input:**

  `model` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

  `dw` valore empirico della statistica *D–W*

- **Formula:**

  `dw`

$$\sum_{i=2}^{n} (e_i - e_{i-1})^2 \,/\, RSS$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> durbin.watson(model = modello)

 lag Autocorrelation D-W Statistic p-value
   1      -0.1116268       1.75205   0.594
 Alternative hypothesis: rho != 0
```

AIC()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell} + 6$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> AIC(object = modello)

[1] 36.61846
```

extractAIC()

- **Package:** stats
- **Input:**

  `fit` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** numero di parametri del modello ed indice *AIC* generalizzato
- **Formula:**

$$2 \qquad n \log(RSS\,/\,n) + 4$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> extractAIC(fit = modello)

[1]  2.000000 -4.720086
```

deviance()

- **Package:** **tt stats**
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** devianza residua
- **Formula:**

$$RSS$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> deviance(object = modello)

[1] 2.689703
```

PRESS()

- **Package:** MPV
- **Input:**

  `x` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** PRESS
- **Formula:**

$$\sum_{i=1}^{n} e_i^2 \,/\, (1 - h_i)^2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> PRESS(x = modello)

[1] 53.41271
```

anova()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** anova di regressione
- **Output:**

  `Df` gradi di libertà

  `Sum Sq` devianze residue

  `Mean Sq` quadrati medi

  `F value` valore empirico della statistica F

  `Pr(>F)` p-value

- **Formula:**

  `Df`

$$1 \qquad n-2$$

  `Sum Sq`

$$RSS_{nullo} - RSS \qquad RSS$$

  `Mean Sq`

$$RSS_{nullo} - RSS \qquad RSS \,/\, (n-2)$$

  `F value`

$$F_{value} = \frac{RSS_{nullo} - RSS}{RSS \,/\, (n-2)} = t^2_{\hat{\beta}_2}$$

  `Pr(>F)`

$$P(F_{1,\, n-2} \geq F_{value})$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> anova(object = modello)
```

```
Analysis of Variance Table

Response: y
          Df Sum Sq Mean Sq F value  Pr(>F)
x          1 3.2686  3.2686  7.2914 0.03556 *
Residuals  6 2.6897  0.4483
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

drop1()

- **Package:** `stats`
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

  `scale` selezione indice AIC oppure Cp

  `test = "F"`

- **Description:** 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`

$$RSS_{nullo} - RSS$$

  `RSS`

$$RSS,\; RSS_{nullo}$$

  `AIC`

`scale = 0`

$$n\,\log\left(RSS\,/\,n\right) + 4,\; n\,\log\left(RSS_{nullo}\,/\,n\right) + 2$$

  `Cp`

`scale =` s^2

$$2,\; \frac{RSS_{nullo}}{RSS\,/\,(n-2)} + 2 - n$$

  `F value`

$$F_{value} = \frac{RSS_{nullo} - RSS}{RSS\,/\,(n-2)} = t^2_{\hat{\beta}_2}$$

  `Pr(F)`

$$P(F_{1,\,n-2} \geq F_{value})$$

- **Example 1:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> drop1(object = modello, scale = 0, test = "F")

Single term deletions

Model:
y ~ x
       Df Sum of Sq     RSS     AIC F value   Pr(F)
<none>              2.6897 -4.7201
x       1    3.2686  5.9583 -0.3573  7.2914 0.03556 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> res <- drop1(object = modello, scale = 0, test = "F")
> res$Df

[1] NA  1

> res$"Sum of Sq"

[1]       NA 3.268597

> res$RSS

[1] 2.689703 5.958300

> res$AIC

[1] -4.7200862 -0.3572507

> res$"F value"

[1]       NA 7.291356

> res$"Pr(F)"

[1]         NA 0.03556412
```

- **Example 2:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> s <- summary.lm(object = modello)$sigma
> drop1(object = modello, scale = s^2, test = "F")

Single term deletions

Model:
y ~ x

scale:  0.4482838

       Df Sum of Sq    RSS     Cp F value   Pr(F)
<none>              2.6897 2.0000
x       1    3.2686 5.9583 7.2914  7.2914 0.03556 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> res <- drop1(object = modello, scale = s^2, test = "F")
> res$Df

[1] NA  1

> res$"Sum of Sq"

[1]       NA 3.268597

> res$RSS

[1] 2.689703 5.958300

> res$Cp

[1] 2.000000 7.291356

> res$"F value"

[1]       NA 7.291356

> res$"Pr(F)"

[1]         NA 0.03556412
```

add1()

- **Package:** stats
- **Input:**

  object modello nullo di regressione lineare semplice pesata

  scope modello di regressione lineare pesata con una variabile esplicativa ed n unità

  scale selezione indice AIC oppure Cp

  test = "F"
- **Description:** 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

  $$RSS_{nullo} - RSS$$

  RSS

  $$RSS_{nullo},\, RSS$$

  AIC

`scale = 0`

$$n \log\left(RSS_{nullo} \,/\, n\right) + 2,\; n \log\left(RSS \,/\, n\right) + 4$$

Cp

`scale =` s^2

$$\frac{RSS_{nullo}}{RSS \,/\, (n-2)} + 2 - n,\; 2$$

F value

$$F_{value} = \frac{RSS_{nullo} - RSS}{RSS \,/\, (n-2)} = t^2_{\hat{\beta}_2}$$

Pr(F)

$$P(F_{1,\, n-2} \geq F_{value})$$

- **Example 1:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> nullo <- lm(formula = y ~ 1, weights = rep(1/n, n))
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> add1(object = nullo, scope = modello, scale = 0, test = "F")

Single term additions

Model:
y ~ 1
       Df Sum of Sq    RSS     AIC F value   Pr(F)
<none>              5.9583 -0.3573
x       1    3.2686 2.6897 -4.7201  7.2914 0.03556 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> res <- add1(object = nullo, scope = modello, scale = 0, test = "F")
> res$Df

[1] NA  1

> res$"Sum of Sq"

[1]       NA 3.268597

> res$RSS

[1] 5.958300 2.689703

> res$AIC

[1] -0.3572507 -4.7200862

> res$"F value"

[1]       NA 7.291356

> res$"Pr(F)"

[1]         NA 0.03556412
```

- **Example 2:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> nullo <- lm(formula = y ~ 1, weights = rep(1/n, n))
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> s <- summary.lm(object = modello)$sigma
> add1(object = nullo, scope = modello, scale = s^2, test = "F")
```

```
Single term additions

Model:
y ~ 1

scale:  0.4482838

       Df Sum of Sq    RSS     Cp F value   Pr(F)
<none>              5.9583 7.2914
x       1    3.2686 2.6897 2.0000  7.2914 0.03556 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> res <- add1(object = nullo, scope = modello, scale = s^2, test = "F")
> res$Df
```

```
[1] NA  1
```

```
> res$"Sum of Sq"
```

```
[1]       NA 3.268597
```

```
> res$RSS
```

```
[1] 5.958300 2.689703
```

```
> res$Cp
```

```
[1] 7.291356 2.000000
```

```
> res$"F value"
```

```
[1]       NA 7.291356
```

```
> res$"Pr(F)"
```

```
[1]         NA 0.03556412
```

15.4 Diagnostica

ls.diag()

- **Package:** stats
- **Input:**

  ls.out modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** 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 delle 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

  $$h_i \quad \forall\, i \,=\, 1,\, 2,\, \dots,\, n$$

  std.res

  $$rstandard_i \quad \forall i \,=\, 1,\, 2,\, \dots,\, n$$

  stud.res

  $$rstudent_i \quad \forall i \,=\, 1,\, 2,\, \dots,\, n$$

  cooks

  $$cd_i \quad \forall\, i \,=\, 1,\, 2,\, \dots,\, n$$

  dfits

  $$rstudent_i \sqrt{\frac{h_i}{1 - h_i}} \quad \forall\, i \,=\, 1,\, 2,\, \dots,\, n$$

  correlation

  $$r_{\hat{\beta}_i\, \hat{\beta}_j} \quad \forall\, i, j \,=\, 1,\, 2$$

  std.err

  $$s_{\hat{\beta}_j} \quad \forall\, j \,=\, 1,\, 2$$

  cov.scaled

  $$s^2\, (X^T\, W^{-1}\, X)^{-1}$$

  cov.unscaled

  $$(X^T\, W^{-1}\, X)^{-1}$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> res <- ls.diag(ls.out = modello)
> res$std.dev

[1] 1.893745
```

```
> res$hat

[1] 0.4350043 0.2701267 0.1284350 0.1945578 0.4684951 0.1733040 0.1355195
[8] 0.1945578

> res$std.res

[1] -2.22897996  0.51181072  1.34601741 -0.04039112 -1.20017856  0.81532985
[7]  0.31550428 -0.15806803

> res$stud.res

[1] -4.90710471  0.47776268  1.47068630 -0.03687690 -1.25680777  0.78929887
[7]  0.29043398 -0.14459710

> res$cooks

[1] 1.9126289653 0.0484739848 0.1334918569 0.0001970407 0.6348329327
[6] 0.0696786009 0.0078023824 0.0030176734

> res$dfits

[1] -4.30575707  0.29065126  0.56456215 -0.01812431 -1.17996116  0.36138726
[7]  0.11499284 -0.07106678

> res$correlation

            (Intercept)          x
(Intercept)   1.0000000 -0.8971215
x            -0.8971215  1.0000000

> res$std.err

                [,1]
(Intercept) 4.286587
x           0.784814

> res$cov.scaled

            (Intercept)          x
(Intercept)   18.374825 -3.0180723
x             -3.018072  0.6159331

> res$cov.unscaled

            (Intercept)          x
(Intercept)   5.1236582 -0.8415629
x            -0.8415629  0.1717475
```

cooks.distance()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> cooks.distance(model = modello)

           1            2            3            4            5            6
1.9126289653 0.0484739848 0.1334918569 0.0001970407 0.6348329327 0.0696786009
           7            8
0.0078023824 0.0030176734
```

rstandard()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> rstandard(model = modello)

          1           2           3           4           5           6
-2.22897996  0.51181072  1.34601741 -0.04039112 -1.20017856  0.81532985
          7           8
 0.31550428 -0.15806803
```

rstandard.lm()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> rstandard.lm(model = modello)

          1           2           3           4           5           6
-2.22897996  0.51181072  1.34601741 -0.04039112 -1.20017856  0.81532985
          7           8
 0.31550428 -0.15806803
```

rstudent.lm()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> rstudent.lm(model = modello)

          1           2           3           4           5           6
-4.90710471  0.47776268  1.47068630 -0.03687690 -1.25680777  0.78929887
          7           8
 0.29043398 -0.14459710
```

lmwork()

- **Package:** MASS
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** 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

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> res <- lmwork(object = modello)
> res$stdedv

[1] 0.66954

> res$stdres

          1           2           3           4           5           6
-2.22897996  0.51181072  1.34601741 -0.04039112 -1.20017856  0.81532985
          7           8
 0.31550428 -0.15806803

> res$studres

          1           2           3           4           5           6
-4.90710471  0.47776268  1.47068630 -0.03687690 -1.25680777  0.78929887
          7           8
 0.29043398 -0.14459710
```

dffits()

- **Package:** stats
- **Input:**

  model modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** dffits
- **Formula:**

$$rstudent_i \sqrt{\frac{h_i}{1 - h_i}} \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> dffits(model = modello)

          1           2           3           4           5           6
-4.30575707  0.29065126  0.56456215 -0.01812431 -1.17996116  0.36138726
          7           8
 0.11499284 -0.07106678
```

covratio()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** covratio
- **Formula:**

$$cr_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> covratio(model = modello)

         1          2          3          4          5          6          7
0.07534912 1.80443448 0.80504974 1.78686556 1.56459066 1.37727804 1.61092794
         8
1.77297867
```

lm.influence()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** 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`

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

  `coefficients`

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = w_i\, e_i\, (1 - h_i)^{-1}\, (X^T\, W^{-1}\, X)^{-1}_j\, X_i^T \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2$$

  `sigma`

$$s_{-i} \quad \forall\, i = 1, 2, \ldots, n$$

  `wt.res`

$$\sqrt{w_i}\, e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> res <- lm.influence(model = modello)
> res$hat
```

```
        1         2         3         4         5         6         7         8
0.4350043 0.2701267 0.1284350 0.1945578 0.4684951 0.1733040 0.1355195 0.1945578
```

```
> res$coefficients
```

```
   (Intercept)            x
1 -2.946804056  0.458130527
2  0.452110031 -0.063325849
3  0.456185994 -0.023446758
4  0.005484663 -0.003293542
5  0.922114131 -0.267715952
6  0.480231536 -0.054685694
7  0.033006665  0.009657123
8  0.021463873 -0.012889065
```

```
> res$sigma
```

```
        1         2         3         4         5         6         7         8
0.3041287 0.7172552 0.6127836 0.7333446 0.6393719 0.6916214 0.7273348 0.7319156
```

```
> res$wt.res
```

```
          1           2           3           4           5           6
-1.12177375  0.29275860  0.84135081 -0.02427055 -0.58583599  0.49634403
          7           8
 0.19640809 -0.09498123
```

weights()

- **Package:** stats
- **Input:**

  object modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** pesi
- **Formula:**

$$w_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> weights(object = modello)
```

```
[1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125
```

weighted.residuals()

- **Package:** stats
- **Input:**

  `obj` modello di regressione lineare pesata con una variabile esplicativa ed n unità
- **Description:** residui pesati
- **Formula:**

$$\sqrt{w_i}\, e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> weighted.residuals(obj = modello)

          1           2           3           4           5           6
-1.12177375  0.29275860  0.84135081 -0.02427055 -0.58583599  0.49634403
          7           8
 0.19640809 -0.09498123
```

residuals.lm()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

  `type = "response" / "pearson"` tipo di residuo
- **Description:** residui
- **Formula:**

`type = "response"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$\sqrt{w_i}\, e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> residuals.lm(object = modello, type = "response")

          1           2           3           4           5           6
-3.17285530  0.82804637  2.37969944 -0.06864749 -1.65699442  1.40387291
          7           8
 0.55552598 -0.26864749

> residuals.lm(object = modello, type = "pearson")

          1           2           3           4           5           6
-1.12177375  0.29275860  0.84135081 -0.02427055 -0.58583599  0.49634403
          7           8
 0.19640809 -0.09498123
```

df.residual()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** gradi di libertà della devianza residua
- **Formula:**

$$n - 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> df.residual(object = modello)

[1] 6
```

hatvalues()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> hatvalues(model = modello)

        1         2         3         4         5         6         7         8
0.4350043 0.2701267 0.1284350 0.1945578 0.4684951 0.1733040 0.1355195 0.1945578
```

dfbeta()

- **Package:** stats
- **Input:**

  `formula` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** dfbeta
- **Formula:**

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = w_i\, e_i\, (1 - h_i)^{-1}\, (X^T\, W^{-1}\, X)^{-1}_j\, X^T_i \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> dfbeta(modello)

   (Intercept)            x
1 -2.946804056  0.458130527
2  0.452110031 -0.063325849
3  0.456185994 -0.023446758
4  0.005484663 -0.003293542
5  0.922114131 -0.267715952
6  0.480231536 -0.054685694
7  0.033006665  0.009657123
8  0.021463873 -0.012889065
```

dfbetas()

- **Package:** stats
- **Input:**

  `formula` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** dfbetas
- **Formula:**

$$\frac{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}{s_{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}} = \frac{w_i\, e_i\, (1 - h_i)^{-1}\, (X^T\, W^{-1}\, X)_j^{-1}\, X_i^T}{s_{-i}\, \sqrt{(X^T\, W^{-1}\, X)_{j,\,j}^{-1}}} \quad \forall i\, = 1, 2, \ldots, n \quad \forall j\, = 1, 2$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> dfbetas(modello)

   (Intercept)           x
1 -4.280591734  3.63485094
2  0.278471258 -0.21304046
3  0.328885485 -0.09232735
4  0.003304089 -0.01083702
5  0.637149075 -1.01035839
6  0.306755388 -0.19079196
7  0.020048284  0.03203820
8  0.012955584 -0.04249278
```

outlier.test()

- **Package:** car
- **Input:**

  `model` modello di regressione lineare pesata con una variabile esplicativa ed n unità

- **Description:** test sugli *outliers*
- **Output:**

  `test` massimo residuo studentizzato assoluto, gradi di libertà, p-value

- **Formula:**

  test
  $$t = \max_i(\,|\,rstudent_i\,|) \quad n-3 \quad p\text{-value} = 2\,P(\,t_{n-3} \leq -|\,t\,|) \qquad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> outlier.test(model = modello)

max|rstudent| = 4.907105, degrees of freedom = 5,
unadjusted p = 0.004446945, Bonferroni p = 0.03557556

Observation: 1

> res <- outlier.test(model = modello)
> res$test

max|rstudent|            df  unadjusted p  Bonferroni p
  4.907104708   5.000000000   0.004446945   0.035575564
```

influence.measures()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con una variabile esplicativa ed n unità
- **Description:** dfbetas, dffits, covratio, distanza di *Cook*, valori di leva
- **Output:**

  `infmat` misure di influenza di dimensione $n \times 6$

  `is.inf` matrice di influenza con valori logici di dimensione $n \times 6$
- **Formula:**

$$DFBETAS_{ij} = \frac{w_i\, e_i\, (1-h_i)^{-1}\, (X^T\, W^{-1}\, X)_j^{-1}\, X_i^T}{s_{-i}\, \sqrt{(X^T\, W^{-1}\, X)^{-1}_{j,\,j}}} \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2$$

$$DFFITS_i = rstudent_i\, \sqrt{\frac{h_i}{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$$

$$COVRATIO_i = (1-h_i)^{-1}\, \left(1 + \frac{rstudent_i^2 - 1}{n-2}\right)^{-2} \quad \forall i = 1, 2, \ldots, n$$

$$COOKD_i = \frac{h_i\, rstandard_i^2}{2\,(1-h_i)} \quad \forall\, i = 1, 2, \ldots, n$$

$$HAT_i = h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x, weights = rep(1/n, n))
> res <- influence.measures(model = modello)
> res$infmat

        dfb.1_       dfb.x       dffit       cov.r       cook.d        hat
1 -4.280591734  3.63485094 -4.30575707 0.07534912 1.9126289653 0.4350043
```

```
2  0.278471258 -0.21304046  0.29065126 1.80443448 0.0484739848 0.2701267
3  0.328885485 -0.09232735  0.56456215 0.80504974 0.1334918569 0.1284350
4  0.003304089 -0.01083702 -0.01812431 1.78686556 0.0001970407 0.1945578
5  0.637149075 -1.01035839 -1.17996116 1.56459066 0.6348329327 0.4684951
6  0.306755388 -0.19079196  0.36138726 1.37727804 0.0696786009 0.1733040
7  0.020048284  0.03203820  0.11499284 1.61092794 0.0078023824 0.1355195
8  0.012955584 -0.04249278 -0.07106678 1.77297867 0.0030176734 0.1945578
```

```
> res$is.inf
```

```
  dfb.1_ dfb.x dffit cov.r cook.d   hat
1   TRUE  TRUE  TRUE FALSE   TRUE FALSE
2  FALSE FALSE FALSE FALSE  FALSE FALSE
3  FALSE FALSE FALSE FALSE  FALSE FALSE
4  FALSE FALSE FALSE FALSE  FALSE FALSE
5  FALSE  TRUE FALSE FALSE  FALSE FALSE
6  FALSE FALSE FALSE FALSE  FALSE FALSE
7  FALSE FALSE FALSE FALSE  FALSE FALSE
8  FALSE FALSE FALSE FALSE  FALSE FALSE
```

- **Note 1:** Il caso i-esimo è influente se $|DFBETAS_{ij}| > 1 \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2$
- **Note 2:** Il caso i-esimo è influente se $|DFFITS_i| > 3 \sqrt{2 \,/\, (n-2)} \quad \forall i = 1, 2, \ldots, n$
- **Note 3:** Il caso i-esimo è influente se $|1 - COVRATIO_i| > 6 \,/\, (n-2) \quad \forall i = 1, 2, \ldots, n$
- **Note 4:** Il caso i-esimo è influente se $P\,(F_{2,n-2} \geq COOKD_i) > 0.5 \quad \forall i = 1, 2, \ldots, n$
- **Note 5:** Il caso i-esimo è influente se $HAT_i > 6 \,/\, n \quad \forall i = 1, 2, \ldots, n$
- **Note 6:** I casi influenti rispetto ad almeno una tra queste misure sono marcati con un asterisco. Corrispondentemente la stessa riga della matrice `is.inf` riporterà almeno un simbolo `TRUE`.

Capitolo 16

Regressione lineare multipla pesata

16.1 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 : $X_i \, = \, (1, \, x_{i1}, \, x_{i2}, \ldots, \, x_{ik-1}) \quad \forall \, i \, = \, 1, \, 2, \, \ldots, \, n$
- vettore numerico positivo dei pesi WLS: $w \, = \, (w_1, \, w_2, \, \ldots, \, w_n)$
- matrice diagonale definita positiva di dimensione $n \times n$: $W \, = \, \text{diag}(w_1^{-1}, \, w_2^{-1}, \, \ldots, \, w_n^{-1})$
- matrice di proiezione di dimensione $n \times n$: $H \, = \, X \, (X^T \, W^{-1} \, X)^{-1} \, X^T \, W^{-1}$
- matrice identità di dimensione $n \times n$: I_n
- devianza residua: $RSS \, = \, \sum_{i=1}^{n} \, w_i \, e_i^2 \, = \, y^T \, W^{-1} \, e \, = \, y^T \, W^{-1} \, (I_n - H) \, y$
- 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: $s_{-i}^2 \, = \, s^2 \, \left(1 + \frac{1 - rstandard_i^2}{n-k-1}\right) \, = \, s^2 \, \left(1 + \frac{rstudent_i^2 - 1}{n-k}\right)^{-1} \quad \forall \, i \, = \, 1, \, 2, \, \ldots, \, n$
- stime WLS: $\hat{\beta} \, = \, (X^T \, W^{-1} \, X)^{-1} \, X^T \, W^{-1} \, y$
- standard error delle stime WLS: $s_{\hat{\beta}} \, = \, s \, \sqrt{\text{diag}((X^T \, W^{-1} \, X)^{-1})}$
- t-values delle stime WLS: $t_{\hat{\beta}} \, = \, \hat{\beta} \, / \, s_{\hat{\beta}}$
- residui: $e \, = \, (I_n - H) \, y$
- residui pesati: $\sqrt{w_i} \, e_i \quad \forall \, i \, = \, 1, \, 2, \, \ldots, \, n$
- residui standard: $rstandard_i \, = \, \frac{e_i}{s \, \sqrt{(1-h_i) \, / \, w_i}} \quad \forall \, i \, = \, 1, \, 2, \, \ldots, \, n$
- residui studentizzati: $rstudent_i \, = \, \frac{e_i}{s_{-i} \, \sqrt{(1-h_i) \, / \, w_i}} \, = \, rstandard_i \, \sqrt{\frac{n-k-1}{n-k-rstandard_i^2}} \quad \forall \, i \, = \, 1, \, 2, \, \ldots, \, n$
- valori adattati: $\hat{y} \, = \, H \, y$
- valori di leva: $h_i \, = \, H_{i,i} \quad \forall \, i \, = \, 1, \, 2, \, \ldots, \, n$
- stime WLS tolta la i-esima unità: $\hat{\beta}_{(-i)} \quad \forall \, i \, = \, 1, \, 2, \, \ldots, \, n$
- correlazione delle stime WLS: $r_{\hat{\beta}_i \, \hat{\beta}_j} \, = \, \frac{s^2 \, (X^T \, W^{-1} \, X)_{i,\,j}^{-1}}{s_{\hat{\beta}_i} \, s_{\hat{\beta}_j}} \quad \forall \, i, j \, = \, 1, \, 2, \, \ldots, \, k$
- 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: $R^2 \, = \, 1 - RSS \, / \, RSS_{nullo} \, = \, 1 - (1 - R_{adj}^2) \, (n-k) \, / \, (n-1)$

- indice di determinazione aggiustato: $R^2_{adj} = 1 - \frac{RSS\,/\,(n-k)}{RSS_{nullo}\,/\,(n-1)} = 1 - \left(1 - R^2\right)\,(n-1)\,/\,(n-k)$
- valore noto dei regressori per la previsione: $x_0^T = (1,\, x_{01},\, x_{02},\, \ldots,\, x_{0k-1})$
- log-verosimiglianza normale: $\hat{\ell} = -n\,(\log(2\,\pi) + \log\,(RSS\,/\,n) + 1 - \sum_{i=1}^n \log(w_i)\,/\,n)\;/\,2$
- 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: $cr_i = (1 - h_i)^{-1}\left(1 + \frac{rstudent_i^2 - 1}{n-k}\right)^{-k} = (1 - h_i)^{-1}\left(\frac{s_{-i}}{s}\right)^{2\,k} \quad \forall i = 1,\, 2,\, \ldots,\, n$

16.2 Stima

lm()

- **Package:** `stats`
- **Input:**

  `formula` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

  `weights` pesi

  `x = TRUE` matrice del modello

  `y = TRUE` variabile dipendente
- **Description:** analisi di regressione lineare pesata
- **Output:**

  `coefficients` stime WLS

  `residuals` residui

  `fitted.values` valori adattati

  `weights` pesi

  `rank` rango della matrice del modello

  `df.residual` gradi di libertà della devianza residua

  `x` matrice del modello

  `y` variabile dipendente
- **Formula:**

  `coefficients`
  $$\hat{\beta}_j \quad \forall\, j = 1,\, 2,\, \ldots,\, k$$

  `residuals`
  $$e_i \quad \forall\, i = 1,\, 2,\, \ldots,\, n$$

  `fitted.values`
  $$\hat{y}_i \quad \forall\, i = 1,\, 2,\, \ldots,\, n$$

  `weights`
  $$w_i \quad \forall\, i = 1,\, 2,\, \ldots,\, n$$

  `rank`
  $$k$$

  `df.residual`
  $$n - k$$

  `x`
  $$X$$

  `y`
  $$y$$
- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n), x = TRUE, y = TRUE)
> modello$coefficients

 (Intercept)           x1           x2           x3
 0.988514333  0.422516384 -0.001737381  0.716029046

> modello$residuals

         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227

> modello$fitted.values

        1         2         3         4         5         6         7         8
 2.453638  5.964158  8.293288  8.102518  8.602437  7.139221  9.569117 10.035623

> modello$weights

[1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125

> modello$rank

[1] 4

> modello$df.residual

[1] 4

> modello$x

  (Intercept)  x1  x2   x3
1           1 1.1 1.2 1.40
2           1 2.3 3.4 5.60
3           1 4.5 5.6 7.56
4           1 6.7 7.5 6.00
5           1 8.9 7.5 5.40
6           1 3.4 6.7 6.60
7           1 5.6 8.6 8.70
8           1 6.7 7.6 8.70
attr(,"assign")
[1] 0 1 2 3

> modello$y

   1    2    3    4    5    6    7    8
1.50 6.40 9.60 8.80 8.86 7.80 8.60 8.60
```

- **Note 1:** Il modello nullo si ottiene con `lm(formula = y ~ 1,weights = w)`.
- **Note 2:** L'istruzione `update(object = y ~ x1 + x2,formula = . ~ . + x3)` è esattamente equivalente a `lm(formula = y ~ x1 + x2 + x3,weights = w)`.

- **Note 3:** In seguito ad una modifica come ad esempio `x1[3] <- 1.2`, conviene adoperare il comando `update(modello)` anziché ripetere `modello <- lm(formula = y ~ x1 + x2 + x3,weights = w)`.
- **Note 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),weights = w)`.
- **Note 5:** Per regressioni polinomiali occorre usare il comando `poly()`. Per un polinomio di quarto grado occorre scrivere `lm(formula = y ~ poly(x,degree = 4,raw = TRUE),weights = w)`.
- **Note 6:** 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),weights = w)`.
- **Note 7:** Il comando uzione `lm(formula = y ~ x1 + x2 + x3,weights=w)` è esattamente equivalente a `lm(formula = y ~ X-1,weights = w)`.
- **Note 8:** Il comando `lm(formula = y ~ x1 + x2 + x3,weights = w)` è esattamente equivalente a `lm(formula = y ~ 1 + x1 + x2 + x3,weights = w)`.

summary.lm()

- **Package:** `stats`
- **Input:**

  `object` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

  `correlation = TRUE` correlazione delle stime WLS

- **Description:** analisi di regressione lineare pesata
- **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 WLS non scalata per σ^2

  `correlation` matrice di correlazione delle stime WLS

- **Formula:**

  `residuals`

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

  `coefficients`

$$\hat{\beta}_j \qquad s_{\hat{\beta}_j} \qquad t_{\hat{\beta}_j} \qquad p\text{-value} = 2\,P(t_{n-k} \leq -\,|\,t_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1, 2, \ldots, k$$

  `sigma`

$$s$$

  `r.squared`

$$R^2$$

  `adj.r.squared`

$$R^2_{adj}$$

  `fstatistic`

$$Fvalue = \frac{(RSS_{nullo} - RSS)\,/\,(k-1)}{RSS\,/\,(n-k)} \qquad k-1 \qquad n-k$$

  `cov.unscaled`

$$(X^T\,W^{-1}\,X)^{-1}$$

```
correlation
```

$$r_{\hat{\beta}_i \, \hat{\beta}_j} \quad \forall i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> res <- summary.lm(object = modello, correlation = TRUE)
> res$residuals

         1          2          3          4          5          6          7
-0.3371620  0.1540936  0.4619923  0.2465971  0.0910624  0.2336206 -0.3426347
         8
-0.5075693

> res$coefficients

                Estimate Std. Error      t value  Pr(>|t|)
(Intercept)  0.988514333  1.4292308  0.691640822 0.5272118
x1           0.422516384  0.3883267  1.088043731 0.3377443
x2          -0.001737381  0.5822146 -0.002984091 0.9977619
x3           0.716029046  0.4068987  1.759723294 0.1532663

> res$sigma

[1] 0.4608596

> res$r.squared

[1] 0.8574147

> res$adj.r.squared

[1] 0.7504757

> res$fstatistic

   value    numdf    dendf
8.017793 3.000000 4.000000

> res$cov.unscaled

            (Intercept)         x1         x2         x3
(Intercept)   9.6176174 -0.4860697  0.2804424 -1.2685405
x1           -0.4860697  0.7099981 -0.8751626  0.3633297
x2            0.2804424 -0.8751626  1.5959854 -0.8947971
x3           -1.2685405  0.3633297 -0.8947971  0.7795344

> res$correlation

            (Intercept)         x1          x2         x3
(Intercept)  1.00000000 -0.1860100  0.07158062 -0.4632900
x1          -0.18600997  1.0000000 -0.82213982  0.4883764
x2           0.07158062 -0.8221398  1.00000000 -0.8022181
x3          -0.46329002  0.4883764 -0.80221810  1.0000000
```

vcov()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con $k - 1$ variabili esplicative ed n unità

- **Description:** matrice di covarianza delle stime WLS
- **Formula:**

$$s^2 \, (X^T \, W^{-1} \, X)^{-1}$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> vcov(object = modello)

            (Intercept)          x1          x2          x3
(Intercept)  2.04270054 -0.10323710  0.05956359 -0.26942727
x1          -0.10323710  0.15079759 -0.18587712  0.07716815
x2           0.05956359 -0.18587712  0.33897378 -0.19004733
x3          -0.26942727  0.07716815 -0.19004733  0.16556652
```

lm.wfit()

- **Package:** stats
- **Input:**

  `x` matrice del modello

  `y` variabile dipendente

  `w` pesi

- **Description:** analisi di regressione lineare pesata
- **Output:**

  `coefficients` stime WLS

  `residuals` residui

  `fitted.values` valori adattati

  `weights` pesi

  `rank` rango della matrice del modello

  `df.residual` gradi di libertà della devianza residua

- **Formula:**

  `coefficients`

$$\hat{\beta}_j \quad \forall \, j = 1, 2, \ldots, k$$

  `residuals`

$$e_i \quad \forall \, i = 1, 2, \ldots, n$$

  `fitted.values`

$$\hat{y}_i \quad \forall \, i = 1, 2, \ldots, n$$

  `weights`

$$w_i \quad \forall \, i = 1, 2, \ldots, n$$

`rank`
k

`df.residual`
$n - k$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> X <- model.matrix(object = modello)
> res <- lm.wfit(x = X, y, w = rep(1/n, n))
> res$coefficients

 (Intercept)           x1           x2           x3
 0.988514333  0.422516384 -0.001737381  0.716029046

> res$residuals

[1] -0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
[8] -1.4356227

> res$fitted.values

[1]  2.453638  5.964158  8.293288  8.102518  8.602437  7.139221  9.569117
[8] 10.035623

> res$weights

[1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125

> res$rank

[1] 4

> res$df.residual

[1] 4
```

lsfit()

- **Package:** stats
- **Input:**

  `x` matrice del modello

  `y` variabile dipendente

  `wt` pesi

  `intercept = FALSE`

- **Description:** analisi di regressione lineare pesata
- **Output:**

  `coefficients` stime WLS

`residuals` residui

`wt` pesi

- **Formula:**

`coefficients`

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

`residuals`

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

`wt`

$$w_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> X <- model.matrix(object = modello)
> res <- lsfit(x = X, y, wt = rep(1/n, n), intercept = FALSE)
> res$coefficients

 (Intercept)           x1           x2           x3
 0.988514333  0.422516384 -0.001737381  0.716029046

> res$residuals

[1] -0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
[8] -1.4356227

> res$wt

[1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125
```

confint()

- **Package:** `stats`
- **Input:**

`object` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

`parm` parametri del modello su cui calcolare l'intervallo di confidenza

`level` livello di confidenza $1-\alpha$

- **Description:** intervallo di confidenza per le stime WLS
- **Formula:**

$$\hat{\beta}_j \mp t_{1-\alpha\,/\,2,\,n-k}\, s_{\hat{\beta}_j} \quad \forall j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> confint(object = modello, parm = c(1, 2, 3), level = 0.95)
```

```
                 2.5 %   97.5 %
(Intercept) -2.9796664 4.956695
x1          -0.6556513 1.500684
x2          -1.6182241 1.614749
```

Confint()

- **Package:** Rcmdr
- **Input:**

  `object` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

  `parm` parametri del modello su cui calcolare l'intervallo di confidenza

  `level` livello di confidenza $1-\alpha$

- **Description:** intervallo di confidenza per le stime WLS
- **Formula:**

$$\hat{\beta}_j \mp t_{1-\alpha\,/\,2,\,n-k}\, s_{\hat{\beta}_j} \quad \forall j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> Confint(object = modello, parm = c(1, 2, 3), level = 0.95)
```

```
                 2.5 %   97.5 %
(Intercept) -2.9796664 4.956695
x1          -0.6556513 1.500684
x2          -1.6182241 1.614749
```

coef()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** stime WLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> coef(object = modello)
```

```
 (Intercept)           x1           x2           x3
 0.988514333  0.422516384 -0.001737381  0.716029046
```

coefficients()

- **Package:** stats
- **Input:**

  object modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** stime WLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> coefficients(object = modello)

 (Intercept)           x1           x2           x3
 0.988514333  0.422516384 -0.001737381  0.716029046
```

coeftest()

- **Package:** lmtest
- **Input:**

  x 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

- **Description:** stima puntuale, standard error, t-value, p-value
- **Formula:**

df = NULL

$$\hat{\beta}_j \qquad s_{\hat{\beta}_j} \qquad t_{\hat{\beta}_j} \qquad p\text{-value} = 2\,P(t_{n-k} \le -\,|\,t_{\hat{\beta}_j}\,|) \qquad \forall\, j = 1, 2, \ldots, k$$

df = Inf

$$\hat{\beta}_j \qquad s_{\hat{\beta}_j} \qquad z_{\hat{\beta}_j} \qquad p\text{-value} = 2\,\Phi\left(-\,|\,z_{\hat{\beta}_j}\,|\right) \qquad \forall\, j = 1, 2, \ldots, k$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> coeftest(x = modello, df = NULL)
```

```
t test of coefficients:

             Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.9885143  1.4292308  0.6916   0.5272
x1           0.4225164  0.3883267  1.0880   0.3377
x2          -0.0017374  0.5822146 -0.0030   0.9978
x3           0.7160290  0.4068987  1.7597   0.1533
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> coeftest(x = modello, df = Inf)

z test of coefficients:

             Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.9885143  1.4292308  0.6916  0.48916
x1           0.4225164  0.3883267  1.0880  0.27658
x2          -0.0017374  0.5822146 -0.0030  0.99762
x3           0.7160290  0.4068987  1.7597  0.07845 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- **Note:** Naturalmente vale che $t_{\hat{\beta}_j} = z_{\hat{\beta}_j} \quad \forall\, j = 1, 2, \ldots, k$.

fitted()

- **Package:** stats
- **Input:**

  object modello di regressione lineare pesata con $k - 1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{y}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> fitted(object = modello)

        1         2         3         4         5         6         7         8
 2.453638  5.964158  8.293288  8.102518  8.602437  7.139221  9.569117 10.035623
```

fitted.values()

- **Package:** stats
- **Input:**

  object modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità
- **Description:** valori adattati
- **Formula:**

$$\hat{y}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> fitted.values(object = modello)

       1         2         3         4         5         6         7         8
2.453638  5.964158  8.293288  8.102518  8.602437  7.139221  9.569117 10.035623
```

predict.lm()

- **Package:** stats
- **Input:**

  object modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

  newdata il valore di x_0

  se.fit = TRUE 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 $1-\alpha$
- **Description:** 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"

$$x_0^T\,\hat{\beta} \qquad x_0^T\,\hat{\beta} \mp t_{1-\alpha\,/\,2,\,df}\; s^* \sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

interval = "prediction"

$$x_0^T\,\hat{\beta} \qquad x_0^T\,\hat{\beta} \mp t_{1-\alpha\,/\,2,\,df}\; s^* \sqrt{1 + x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

se.fit

$$s^* \sqrt{x_0^T \, (X^T \, W^{-1} \, X)^{-1} \, x_0}$$

df

$$df \; = n - k$$

residual.scale

$$s^*$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> x0 <- c(1, 1.3, 2.1, 2.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat

[1] 3.181004

> new <- data.frame(x1 = 1.3, x2 = 2.1, x3 = 2.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> W <- diag(1/rep(1/n, n))
> lower <- yhat - qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     solve(W) %*% X) %*% x0)
> upper <- yhat + qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     solve(W) %*% X) %*% x0)
> c(yhat, lower, upper)

[1] 3.181004 1.200204 5.161803

> res <- predict.lm(object = modello, newdata = new, se.fit = TRUE,
+     scale = s, df = Inf, interval = "confidence", level = 0.95)
> res$fit

       fit      lwr      upr
1 3.181004 1.200204 5.161803

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% solve(W) %*%
+     X) %*% x0))
> se.fit

[1] 1.010631

> res$se.fit

[1] 1.010631

> s

[1] 0.4608596

> res$residual.scale

[1] 0.4608596
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> x0 <- c(1, 1.3, 2.1, 2.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat

[1] 3.181004

> new <- data.frame(x1 = 1.3, x2 = 2.1, x3 = 2.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> W <- diag(1/rep(1/n, n))
> lower <- yhat - qt(1 - 0.05/2, df = n - k) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% solve(W) %*% X) %*% x0)
> upper <- yhat + qt(1 - 0.05/2, df = n - k) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% solve(W) %*% X) %*% x0)
> c(yhat, lower, upper)

[1] 3.18100394 0.09706736 6.26494051

> res <- predict.lm(object = modello, newdata = new, se.fit = TRUE,
+     interval = "prediction", level = 0.95)
> res$fit

       fit        lwr     upr
1 3.181004 0.09706736 6.26494

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% solve(W) %*%
+     X) %*% x0))
> se.fit

[1] 1.010631

> res$se.fit

[1] 1.010631

> s

[1] 0.4608596

> res$residual.scale

[1] 0.4608596
```

- **Note 1:** Per il calcolo dell'intervallo classico di confidenza o previsione impostare i parametri `df = n - k` e `scale = summary.lm(object = modello)$sigma`.
- **Note 2:** Per il calcolo dell'intervallo asintotico di confidenza o previsione impostare i parametri `df = Inf` e `scale = summary.lm(object = modello)$sigma`.

predict()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` 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 $1-\alpha$

- **Description:** 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"`

  $$x_0^T\,\hat{\beta} \qquad x_0^T\,\hat{\beta} \mp t_{1-\alpha\,/\,2,\,df}\; s^*\,\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

  `interval = "prediction"`

  $$x_0^T\,\hat{\beta} \qquad x_0^T\,\hat{\beta} \mp t_{1-\alpha\,/\,2,\,df}\; s^*\,\sqrt{1+x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

  `se.fit`

  $$s^*\,\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

  `df`

  $$df = n-k$$

  `residual.scale`

  $$s^*$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> x0 <- c(1, 1.3, 2.1, 2.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat

[1] 3.181004

> new <- data.frame(x1 = 1.3, x2 = 2.1, x3 = 2.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> W <- diag(1/rep(1/n, n))
> lower <- yhat - qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     solve(W) %*% X) %*% x0)
> upper <- yhat + qnorm(1 - 0.05/2) * s * sqrt(t(x0) %*% solve(t(X) %*%
+     solve(W) %*% X) %*% x0)
> c(yhat, lower, upper)
```

```
[1] 3.181004 1.200204 5.161803
```

```
> res <- predict(object = modello, newdata = new, se.fit = TRUE,
+     scale = s, df = Inf, interval = "confidence", level = 0.95)
> res$fit
```

```
       fit      lwr      upr
1 3.181004 1.200204 5.161803
```

```
> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% solve(W) %*%
+     X) %*% x0))
> se.fit
```

```
[1] 1.010631
```

```
> res$se.fit
```

```
[1] 1.010631
```

```
> s
```

```
[1] 0.4608596
```

```
> res$residual.scale
```

```
[1] 0.4608596
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> x0 <- c(1, 1.3, 2.1, 2.3)
> yhat <- as.numeric(t(x0) %*% coef(object = modello))
> yhat
```

```
[1] 3.181004
```

```
> new <- data.frame(x1 = 1.3, x2 = 2.1, x3 = 2.3)
> s <- summary.lm(object = modello)$sigma
> X <- model.matrix(object = modello)
> W <- diag(1/rep(1/n, n))
> lower <- yhat - qt(1 - 0.05/2, df = n - k) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% solve(W) %*% X) %*% x0)
> upper <- yhat + qt(1 - 0.05/2, df = n - k) * s * sqrt(1 + t(x0) %*%
+     solve(t(X) %*% solve(W) %*% X) %*% x0)
> c(yhat, lower, upper)
```

```
[1] 3.18100394 0.09706736 6.26494051
```

```
> res <- predict(object = modello, newdata = new, se.fit = TRUE,
+     interval = "prediction", level = 0.95)
> res$fit
```

```
       fit        lwr      upr
1 3.181004 0.09706736 6.26494

> se.fit <- as.numeric(s * sqrt(t(x0) %*% solve(t(X) %*% solve(W) %*%
+     X) %*% x0))
> se.fit

[1] 1.010631

> res$se.fit

[1] 1.010631

> s

[1] 0.4608596

> res$residual.scale

[1] 0.4608596
```

- **Note 1:** Per il calcolo dell'intervallo classico di confidenza o previsione impostare i parametri `df = n - k` e `scale = summary.lm(object = modello)$sigma`.
- **Note 2:** Per il calcolo dell'intervallo asintotico di confidenza o previsione impostare i parametri `df = Inf` e `scale = summary.lm(object = modello)$sigma`.

linear.hypothesis()

- **Package:** car
- **Input:**

  `model` 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

- **Description:** 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

$$RSS \qquad RSS + \left(b - C\,\hat{\beta}\right)^T \left[C\,\left(X^T\,W^{-1}\,X\right)^{-1}\,C^T\right]^{-1}\left(b - C\,\hat{\beta}\right)$$

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)

$$P(F_{q,\,n-k} \geq Fvalue)$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> W <- diag(1/rep(1/n, n))
> C <- matrix(c(1, 3, 5, 2.3, 2, 4, 1.1, 4.3), nrow = 2, ncol = 4,
+     byrow = TRUE)
> C

     [,1] [,2] [,3] [,4]
[1,]    1    3  5.0  2.3
[2,]    2    4  1.1  4.3

> b <- c(1.1, 2.3)
> b

[1] 1.1 2.3

> q <- 2
> c(n - k, n - k + q)

[1] 4 6

> linear.hypothesis(model = modello, hypothesis.matrix = C, rhs = b)$Res.Df

[1] 4 6

> X <- model.matrix(object = modello)
> RSS <- sum(weighted.residuals(obj = modello)^2)
> beta <- coefficients(object = modello)
> CSS <- as.numeric(t(b - C %*% beta) %*% solve(C %*% solve(t(X) %*%
+     solve(W) %*% X) %*% t(C)) %*% (b - C %*% beta))
> c(RSS, RSS + CSS)

[1] 0.8495662 2.2459829

> linear.hypothesis(model = modello, hypothesis.matrix = C, rhs = b)$RSS

[1] 0.8495662 2.2459829
```

```
> -q

[1] -2

> linear.hypothesis(model = modello, hypothesis.matrix = C, rhs = b)$Df

[1] NA -2

> -CSS

[1] -1.396417

> linear.hypothesis(model = modello, hypothesis.matrix = C, rhs = b)$"Sum of Sq"

[1]        NA -1.396417

> Fvalue <- (CSS/q)/(RSS/(n - k))
> Fvalue

[1] 3.287364

> linear.hypothesis(model = modello, hypothesis.matrix = C, rhs = b)$F

[1]       NA 3.287364

> 1 - pf(Fvalue, df1 = q, df2 = n - k)

[1] 0.1430808

> linear.hypothesis(model = modello, hypothesis.matrix = C, rhs = b)$"Pr(>F)"

[1]        NA 0.1430808
```

lht()

- **Package:** car
- **Input:**

  `model` 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

- **Description:** 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`

$$RSS \qquad RSS + \left(b - C\,\hat{\beta}\right)^T \left[C\,\left(X^T\,W^{-1}\,X\right)^{-1}\,C^T\right]^{-1}\left(b - C\,\hat{\beta}\right)$$

`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)`

$$P(F_{q,\,n-k} \geq Fvalue)$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> W <- diag(1/rep(1/n, n))
> C <- matrix(c(1, 3, 5, 2.3, 2, 4, 1.1, 4.3), nrow = 2, ncol = 4,
+     byrow = TRUE)
> C

     [,1] [,2] [,3] [,4]
[1,]    1    3  5.0  2.3
[2,]    2    4  1.1  4.3

> b <- c(1.1, 2.3)
> b

[1] 1.1 2.3

> q <- 2
> c(n - k, n - k + q)

[1] 4 6

> lht(model = modello, hypothesis.matrix = C, rhs = b)$Res.Df

[1] 4 6

> X <- model.matrix(object = modello)
> RSS <- sum(weighted.residuals(obj = modello)^2)
> beta <- coefficients(object = modello)
> CSS <- as.numeric(t(b - C %*% beta) %*% solve(C %*% solve(t(X) %*%
+     solve(W) %*% X) %*% t(C)) %*% (b - C %*% beta))
> c(RSS, RSS + CSS)
```

```
[1] 0.8495662 2.2459829

> lht(model = modello, hypothesis.matrix = C, rhs = b)$RSS

[1] 0.8495662 2.2459829

> -q

[1] -2

> lht(model = modello, hypothesis.matrix = C, rhs = b)$Df

[1] NA -2

> -CSS

[1] -1.396417

> lht(model = modello, hypothesis.matrix = C, rhs = b)$"Sum of Sq"

[1]        NA -1.396417

> Fvalue <- (CSS/q)/(RSS/(n - k))
> Fvalue

[1] 3.287364

> lht(model = modello, hypothesis.matrix = C, rhs = b)$F

[1]       NA 3.287364

> 1 - pf(Fvalue, df1 = q, df2 = n - k)

[1] 0.1430808

> lht(model = modello, hypothesis.matrix = C, rhs = b)$"Pr(>F)"

[1]        NA 0.1430808
```

cov2cor()

- **Package:** stats
- **Input:**

  V matrice di covarianza delle stime WLS di dimensione $k \times k$

- **Description:** converte la matrice di covarianza nella matrice di correlazione
- **Formula:**

$$r_{\hat{\beta}_i \, \hat{\beta}_j} \quad \forall \, i, j \, = \, 1, \, 2, \, \ldots, \, k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> V <- vcov(object = modello)
> cov2cor(V)

            (Intercept)         x1          x2         x3
(Intercept)  1.00000000 -0.1860100  0.07158062 -0.4632900
x1          -0.18600997  1.0000000 -0.82213982  0.4883764
x2           0.07158062 -0.8221398  1.00000000 -0.8022181
x3          -0.46329002  0.4883764 -0.80221810  1.0000000
```

16.3 Adattamento

logLik()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** log-verosimiglianza normale
- **Formula:**

$$\hat{\ell}$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> logLik(object = modello)

'log Lik.' -10.69939 (df=5)
```

durbin.watson()

- **Package:** car
- **Input:**

  `model` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

  `dw` valore empirico della statistica *D–W*

- **Formula:**

dw
$$\sum_{i=2}^{n} (e_i - e_{i-1})^2 \,/\, RSS$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> durbin.watson(model = modello)$dw

[1] 0.9255503
```

AIC()

- **Package:** stats
- **Input:**

  object modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell} + 2\,(k+1)$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> AIC(object = modello)

[1] 31.39878
```

BIC()

- **Package:** nlme
- **Input:**

  object modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** indice *BIC*
- **Formula:**

$$-2\,\hat{\ell} + (k+1)\,\log(n)$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> BIC(object = modello)

[1] 31.79599
```

extractAIC()

- **Package:** stats
- **Input:**

  `fit` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** numero di parametri del modello ed indice *AIC* generalizzato
- **Formula:**

$$k \qquad n\log(RSS\,/\,n)\,+2\,k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> extractAIC(fit = modello)

[1]  4.000000 -9.939768
```

deviance()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** devianza residua
- **Formula:**

$$RSS$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> deviance(object = modello)

[1] 0.8495662
```

PRESS()

- **Package:** MPV
- **Input:**

  `x` modello di regressione lineare pesata con $k - 1$ variabili esplicative ed n unità

- **Description:** PRESS
- **Formula:**

$$\sum_{i=1}^{n} e_i^2 \,/\, (1 - h_i)^2$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> PRESS(x = modello)

[1] 35.00228
```

drop1()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con $k - 1$ variabili esplicative ed n unità

  `scale` selezione indice AIC oppure Cp

  `test = "F"`

- **Description:** 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

$$\boxed{\texttt{scale = 0}}$$

$$n \log\left(RSS \,/\, n\right) + 2\,k,\; n \log\left(RSS_{-x_j} \,/\, n\right) + 2\,(k-1) \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

Cp

$$\boxed{\texttt{scale = } s^2}$$

$$k,\; \frac{RSS_{-x_j}}{RSS \,/\, (n-k)} + 2\,(k-1) - n \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

F value

$$F_j \,=\, \frac{RSS_{-x_j} - RSS}{RSS \,/\, (n-k)} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

Pr(F)

$$P(F_{1,\, n-k} \geq F_j) \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> drop1(object = modello, scale = 0, test = "F")

Single term deletions

Model:
y ~ x1 + x2 + x3
       Df Sum of Sq      RSS      AIC   F value  Pr(F)
<none>                0.8496  -9.9398
x1      1    0.2514   1.1010  -9.8658    1.1838 0.3377
x2      1 1.891e-06   0.8496 -11.9398 8.905e-06 0.9978
x3      1    0.6577   1.5073  -7.3532    3.0966 0.1533

> res <- drop1(object = modello, scale = 0, test = "F")
> res$Df

[1] NA  1  1  1

> res$"Sum of Sq"

[1]           NA 2.514374e-01 1.891304e-06 6.576972e-01

> res$RSS

[1] 0.8495662 1.1010036 0.8495680 1.5072633

> res$AIC

[1]  -9.939768  -9.865756 -11.939750  -7.353167

> res$"F value"

[1]           NA 1.183839e+00 8.904801e-06 3.096626e+00

> res$"Pr(F)"
```

```
[1]        NA 0.3377443 0.9977619 0.1532663
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> s <- summary.lm(object = modello)$sigma
> drop1(object = modello, scale = s^2, test = "F")

Single term deletions

Model:
y ~ x1 + x2 + x3

scale:  0.2123915

       Df Sum of Sq     RSS     Cp   F value  Pr(F)
<none>                0.84957 4.0000
x1      1   0.25144 1.10100 3.1838    1.1838 0.3377
x2      1 1.891e-06 0.84957 2.0000 8.905e-06 0.9978
x3      1   0.65770 1.50726 5.0966    3.0966 0.1533

> res <- drop1(object = modello, scale = s^2, test = "F")
> res$Df

[1] NA  1  1  1

> res$"Sum of Sq"

[1]           NA 2.514374e-01 1.891304e-06 6.576972e-01

> res$RSS

[1] 0.8495662 1.1010036 0.8495680 1.5072633

> res$Cp

[1] 4.000000 3.183839 2.000009 5.096626

> res$"F value"

[1]           NA 1.183839e+00 8.904801e-06 3.096626e+00

> res$"Pr(F)"

[1]        NA 0.3377443 0.9977619 0.1532663
```

add1()

- **Package:** stats
- **Input:**

  `object` modello nullo di regressione lineare pesata

  `scope` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

  `scale` selezione indice AIC oppure Cp

  `test = "F"`

- **Description:** 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`

$$RSS_{nullo} - RSS_{x_j} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  dove 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 = 0`

$$n \,\log\left(RSS_{nullo} \,/\, n\right) + 2,\; n \,\log\left(RSS_{x_j} \,/\, n\right) + 4 \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  `Cp`

  `scale =` s^2

$$\frac{RSS_{nullo}}{RSS \,/\, (n-k)} + 2 - n,\; \frac{RSS_{x_j}}{RSS \,/\, (n-k)} + 4 - n \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  `F value`

$$F_j \,=\, \frac{RSS_{nullo} - RSS_{x_j}}{RSS_{x_j} \,/\, (n-2)} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  `Pr(F)`

$$P(F_{1,\, n-2} \geq F_j) \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> nullo <- lm(formula = y ~ 1, weights = rep(1/n, n))
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> add1(object = nullo, scope = modello, scale = 0, test = "F")
```

```
Single term additions

Model:
y ~ 1
       Df Sum of Sq     RSS     AIC F value    Pr(F)
<none>              5.9583 -0.3573
x1      1    3.2686  2.6897 -4.7201  7.2914 0.035564 *
x2      1    4.4365  1.5218 -9.2762 17.4911 0.005799 **
x3      1    4.3364  1.6219 -8.7667 16.0418 0.007077 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> res <- add1(object = nullo, scope = modello, scale = 0, test = "F")
> res$Df

[1] NA  1  1  1

> res$"Sum of Sq"

[1]       NA 3.268597 4.436456 4.336392

> res$RSS

[1] 5.958300 2.689703 1.521844 1.621908

> res$AIC

[1] -0.3572507 -4.7200862 -9.2761525 -8.7667043

> res$"F value"

[1]        NA  7.291356 17.491113 16.041811

> res$"Pr(F)"

[1]          NA 0.035564122 0.005799048 0.007076764
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> nullo <- lm(formula = y ~ 1, weights = rep(1/n, n))
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> s <- summary.lm(object = modello)$sigma
> add1(object = nullo, scope = modello, scale = s^2, test = "F")

Single term additions

Model:
y ~ 1

scale:  0.2123915

       Df Sum of Sq    RSS      Cp F value    Pr(F)
<none>             5.9583 22.0534
x1      1    3.2686 2.6897  8.6639  7.2914 0.035564 *
x2      1    4.4365 1.5218  3.1653 17.4911 0.005799 **
x3      1    4.3364 1.6219  3.6364 16.0418 0.007077 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> res <- add1(object = nullo, scope = modello, scale = s^2, test = "F")
> res$Df

[1] NA  1  1  1

> res$"Sum of Sq"

[1]       NA 3.268597 4.436456 4.336392

> res$RSS

[1] 5.958300 2.689703 1.521844 1.621908

> res$Cp

[1] 22.053378  8.663889  3.165274  3.636408

> res$"F value"

[1]        NA  7.291356 17.491113 16.041811

> res$"Pr(F)"

[1]          NA 0.035564122 0.005799048 0.007076764
```

leaps()

- **Package:** leaps
- **Input:**

  x matrice del modello priva della prima colonna (intercetta) di dimensione $n \times (h-1)$

  y variabile dipendente

  wt vettore positivo dei pesi di dimensione n

  method = "r2" / "adjr2" / "Cp" indice R^2, R^2_{adj}, C_p

  nbest = 1
- **Description:** Best Subsets
- **Output:**

  which variabili selezionate

  size numero di parametri

  r2 / adjr2 / Cp indice R^2, R^2_{adj}, C_p
- **Formula:**

  size

$$k_j \quad \forall\, j = 1, 2, \ldots, h-1$$

  r2

  `method = "r2"`

$$R^2_j \quad \forall\, j = 1, 2, \ldots, h-1$$

  R^2_j rappresenta il massimo R^2 tra i $\binom{h-1}{j}$ modelli di regressione con j variabili esplicative oppure k_j parametri.

  adjr2

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	$k_j = j+1$	$\binom{h-1}{j}$
.	.	.
.	.	.
$h-1$	$k_{h-1} = h$	$\binom{h-1}{h-1}$

`method = "adjr2"`

$$R^2_{adj\,j} = 1 - \frac{RSS\,/\,(n-k_j)}{RSS_{nullo}\,/\,(n-1)}$$
$$= \frac{1-k_j}{n-k_j} + \frac{n-1}{n-k_j}\,R^2_j \qquad \forall\, j = 1, 2, \ldots, h-1$$

$R^2_{adj\,j}$ rappresenta il massimo R^2_{adj} tra i $\binom{h-1}{j}$ modelli di regressione con j variabili esplicative oppure k_j parametri.

Cp

`method = "Cp"`

$$Cp_j = (n-k_{h-1})\,\frac{1-R^2_j}{1-R^2_{h-1}} + 2\,k_j - n$$
$$= \left(\frac{n-k_{h-1}}{1-R^2_{h-1}} + 2\,k_j - n\right) - \frac{n-k_{h-1}}{1-R^2_{h-1}}\,R^2_j \qquad \forall\, j = 1, 2, \ldots, h-1$$

Cp_j rappresenta il minimo Cp tra i $\binom{h-1}{j}$ modelli di regressione con j variabili esplicative oppure k_j parametri.

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> X <- model.matrix(object = modello)
> A <- X[, -1]
> leaps(x = A, y, wt = rep(1/n, n), method = "r2", nbest = 1)

$which
      1     2     3
1 FALSE  TRUE FALSE
2  TRUE FALSE  TRUE
3  TRUE  TRUE  TRUE

$label
[1] "(Intercept)" "1"           "2"           "3"

$size
[1] 2 3 4
```

```
$r2
[1] 0.7445843 0.8574144 0.8574147

> res <- leaps(x = A, y, wt = rep(1/n, n), method = "r2", nbest = 1)
> res$which

      1     2     3
1 FALSE  TRUE FALSE
2  TRUE FALSE  TRUE
3  TRUE  TRUE  TRUE

> res$size

[1] 2 3 4

> res$r2

[1] 0.7445843 0.8574144 0.8574147
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> X <- model.matrix(object = modello)
> A <- X[, -1]
> leaps(x = A, y, wt = rep(1/n, n), method = "adjr2", nbest = 1)

$which
      1     2     3
1 FALSE  TRUE FALSE
2  TRUE FALSE  TRUE
3  TRUE  TRUE  TRUE

$label
[1] "(Intercept)" "1"           "2"           "3"

$size
[1] 2 3 4

$adjr2
[1] 0.7020150 0.8003801 0.7504757

> res <- leaps(x = A, y, wt = rep(1/n, n), method = "adjr2", nbest = 1)
> res$which

      1     2     3
1 FALSE  TRUE FALSE
2  TRUE FALSE  TRUE
3  TRUE  TRUE  TRUE

> res$size

[1] 2 3 4
```

```
> res$adjr2

[1] 0.7020150 0.8003801 0.7504757
```

- **Example 3:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> X <- model.matrix(object = modello)
> A <- X[, -1]
> leaps(x = A, y, wt = rep(1/n, n), method = "Cp", nbest = 1)

$which
      1     2     3
1 FALSE  TRUE FALSE
2  TRUE FALSE  TRUE
3  TRUE  TRUE  TRUE

$label
[1] "(Intercept)" "1"           "2"           "3"

$size
[1] 2 3 4

$Cp
[1] 3.165274 2.000009 4.000000


> res <- leaps(x = A, y, wt = rep(1/n, n), method = "Cp", nbest = 1)
> res$which

      1     2     3
1 FALSE  TRUE FALSE
2  TRUE FALSE  TRUE
3  TRUE  TRUE  TRUE

> res$size

[1] 2 3 4

> res$Cp

[1] 3.165274 2.000009 4.000000
```

- **Note 1:** Tutti i modelli contengono l'intercetta.
- **Note 2:** $R^2_{adj\, j}$ è una trasformazione lineare crescente di $R^2_j \quad \forall\, j = 1, 2, \ldots, h-1$.
- **Note 3:** Cp_j è una trasformazione lineare decrescente di $R^2_j \quad \forall\, j = 1, 2, \ldots, h-1$.

16.4 Diagnostica

ls.diag()

- **Package:** stats
- **Input:**

  `ls.out` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità
- **Description:** 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 delle 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`
$$h_i \quad \forall\, i = 1, 2, \ldots, n$$
  `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`
$$(X^T\,W^{-1}\,X)^{-1}$$
- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> res <- ls.diag(ls.out = modello)
> res$std.dev
```

```
[1] 1.303508

> res$hat

[1] 0.7695906 0.4163361 0.3791092 0.3154744 0.7283511 0.5539241 0.4302463
[8] 0.4069682

> res$std.res

[1] -1.5241225  0.4376576  1.2722093  0.6467323  0.3791111  0.7589935 -0.9849613
[8] -1.4301703

> res$stud.res

[1] -2.0384846  0.3884371  1.4278921  0.5918863  0.3343822  0.7104546 -0.9800972
[8] -1.7718134

> res$cooks

[1] 1.93972080 0.03415783 0.24706215 0.04819074 0.09633983 0.17883712 0.18315058
[8] 0.35091186

> res$dfits

[1] -3.7255223  0.3280660  1.1157578  0.4018144  0.5475321  0.7916935 -0.8516950
[8] -1.4677742

> res$correlation

            (Intercept)         x1          x2         x3
(Intercept)  1.00000000 -0.1860100  0.07158062 -0.4632900
x1          -0.18600997  1.0000000 -0.82213982  0.4883764
x2           0.07158062 -0.8221398  1.00000000 -0.8022181
x3          -0.46329002  0.4883764 -0.80221810  1.0000000

> res$std.err

                [,1]
(Intercept) 4.042475
x1          1.098354
x2          1.646751
x3          1.150883

> res$cov.scaled

            (Intercept)         x1         x2         x3
(Intercept)  16.3416044 -0.8258968  0.4765087 -2.1554182
x1           -0.8258968  1.2063807 -1.4870170  0.6173452
x2            0.4765087 -1.4870170  2.7117903 -1.5203786
x3           -2.1554182  0.6173452 -1.5203786  1.3245321

> res$cov.unscaled

            (Intercept)         x1         x2         x3
(Intercept)   9.6176174 -0.4860697  0.2804424 -1.2685405
x1           -0.4860697  0.7099981 -0.8751626  0.3633297
x2            0.2804424 -0.8751626  1.5959854 -0.8947971
x3           -1.2685405  0.3633297 -0.8947971  0.7795344
```

cooks.distance()

- **Package:** stats
- **Input:**

  model modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> cooks.distance(model = modello)

         1          2          3          4          5          6          7
1.93972080 0.03415783 0.24706215 0.04819074 0.09633983 0.17883712 0.18315058
         8
0.35091186
```

cookd()

- **Package:** car
- **Input:**

  model modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> cookd(model = modello)

         1          2          3          4          5          6          7
1.93972080 0.03415783 0.24706215 0.04819074 0.09633983 0.17883712 0.18315058
         8
0.35091186
```

rstandard()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> rstandard(model = modello)

         1          2          3          4          5          6          7
-1.5241225  0.4376576  1.2722093  0.6467323  0.3791111  0.7589935 -0.9849613
         8
-1.4301703
```

rstandard.lm()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> rstandard.lm(model = modello)

         1          2          3          4          5          6          7
-1.5241225  0.4376576  1.2722093  0.6467323  0.3791111  0.7589935 -0.9849613
         8
-1.4301703
```

stdres()

- **Package:** MASS
- **Input:**

  object modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> stdres(object = modello)

         1          2          3          4          5          6          7
-1.5241225  0.4376576  1.2722093  0.6467323  0.3791111  0.7589935 -0.9849613
         8
-1.4301703
```

rstudent()

- **Package:** stats
- **Input:**

  model modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> rstudent(model = modello)

         1          2          3          4          5          6          7
-2.0384846  0.3884371  1.4278921  0.5918863  0.3343822  0.7104546 -0.9800972
         8
-1.7718134
```

rstudent.lm()

- **Package:** stats
- **Input:**

  model modello di regressione lineare pesata con $k - 1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> rstudent.lm(model = modello)

         1          2          3          4          5          6          7
-2.0384846  0.3884371  1.4278921  0.5918863  0.3343822  0.7104546 -0.9800972
         8
-1.7718134
```

studres()

- **Package:** MASS
- **Input:**

  object modello di regressione lineare pesata con $k - 1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> studres(object = modello)

         1          2          3          4          5          6          7
-2.0384846  0.3884371  1.4278921  0.5918863  0.3343822  0.7104546 -0.9800972
         8
-1.7718134
```

lmwork()

- **Package:** MASS
- **Input:**

  `object` modello di regressione lineare pesata con $k - 1$ variabili esplicative ed n unità

- **Description:** diagnostica di regressione
- **Output:**

  `stdedv` stima di σ

  `stdres` residui standard

  `studres` residui studentizzati

- **Formula:**

  `stdedv`

  $$s$$

  `stdres`

  $$rstandard_i \quad \forall i = 1, 2, \dots, n$$

  `studres`

  $$rstudent_i \quad \forall i = 1, 2, \dots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> res <- lmwork(object = modello)
> res$stdedv

[1] 0.4608596

> res$stdres

         1          2          3          4          5          6          7
-1.5241225  0.4376576  1.2722093  0.6467323  0.3791111  0.7589935 -0.9849613
         8
-1.4301703

> res$studres

         1          2          3          4          5          6          7
-2.0384846  0.3884371  1.4278921  0.5918863  0.3343822  0.7104546 -0.9800972
         8
-1.7718134
```

dffits()

- **Package:** stats
- **Input:**

  model modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** dffits
- **Formula:**

$$rstudent_i \sqrt{\frac{h_i}{1 - h_i}} \quad \forall\, i \,=\, 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> dffits(model = modello)

         1          2          3          4          5          6          7
-3.7255223  0.3280660  1.1157578  0.4018144  0.5475321  0.7916935 -0.8516950
         8
-1.4677742
```

covratio()

- **Package:** stats
- **Input:**

  model modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** covratio
- **Formula:**

$$cr_i \quad \forall i \,=\, 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> covratio(model = modello)

         1          2          3          4          5          6          7
 0.4238374  4.4498753  0.6395729  2.9682483 10.0502975  3.8036903  1.8260516
         8
 0.3038647
```

lm.influence()

- **Package:** stats
- **Input:**

  model modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità
- **Description:** 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

$$h_i \quad \forall i = 1, 2, \ldots, n$$

  coefficients

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = w_i\, e_i\, (1 - h_i)^{-1}\, (X^T\, W^{-1}\, X)^{-1}_j\, X_i^T \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

  sigma

$$s_{-i} \quad \forall i = 1, 2, \ldots, n$$

  wt.res

$$\sqrt{w_i}\, e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> lm.influence(model = modello)

$hat
        1         2         3         4         5         6         7         8
0.7695906 0.4163361 0.3791092 0.3154744 0.7283511 0.5539241 0.4302463 0.4069682

$coefficients
  (Intercept)          x1          x2          x3
1 -3.95445343  0.12758388  0.01022818  0.44042192
2  0.21929134  0.01923025 -0.12292616  0.08309302
3 -0.15505077  0.14594807 -0.39064531  0.32853997
4  0.10864633 -0.01436987  0.12965355 -0.11055404
5  0.06456839  0.14591697 -0.04391330 -0.06357315
6  0.27248353 -0.28472521  0.38742501 -0.16358023
7  0.36758841  0.18614884 -0.28071294  0.03129723
8  0.76981755 -0.23622669  0.37474061 -0.34716366

$sigma
        1         2         3         4         5         6         7         8
0.3445728 0.5192571 0.4106121 0.5035642 0.5225068 0.4923459 0.4631468 0.3719961

$wt.res
         1          2          3          4          5          6          7
-0.3371620  0.1540936  0.4619923  0.2465971  0.0910624  0.2336206 -0.3426347
         8
-0.5075693
```

influence()

- **Package:** stats
- **Input:**

  model modello di regressione lineare pesata con $k - 1$ variabili esplicative ed n unità

- **Description:** 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

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

  coefficients

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = w_i\, e_i\, (1 - h_i)^{-1}\, (X^T\, W^{-1}\, X)_j^{-1}\, X_i^T \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

  sigma

$$s_{-i} \quad \forall\, i = 1, 2, \ldots, n$$

  wt.res

$$\sqrt{w_i}\, e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> influence(model = modello)

$hat
        1         2         3         4         5         6         7         8
0.7695906 0.4163361 0.3791092 0.3154744 0.7283511 0.5539241 0.4302463 0.4069682

$coefficients
  (Intercept)          x1          x2          x3
1 -3.95445343  0.12758388  0.01022818  0.44042192
2  0.21929134  0.01923025 -0.12292616  0.08309302
3 -0.15505077  0.14594807 -0.39064531  0.32853997
4  0.10864633 -0.01436987  0.12965355 -0.11055404
5  0.06456839  0.14591697 -0.04391330 -0.06357315
6  0.27248353 -0.28472521  0.38742501 -0.16358023
7  0.36758841  0.18614884 -0.28071294  0.03129723
8  0.76981755 -0.23622669  0.37474061 -0.34716366

$sigma
        1         2         3         4         5         6         7         8
0.3445728 0.5192571 0.4106121 0.5035642 0.5225068 0.4923459 0.4631468 0.3719961

$wt.res
         1          2          3          4          5          6          7
-0.3371620  0.1540936  0.4619923  0.2465971  0.0910624  0.2336206 -0.3426347
         8
-0.5075693
```

weights()

- **Package:** stats
- **Input:**

  object modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** pesi
- **Formula:**

$$w_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> weights(object = modello)

[1] 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125
```

weighted.residuals()

- **Package:** stats
- **Input:**

  obj modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** residui pesati
- **Formula:**

$$\sqrt{w_i}\, e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> weighted.residuals(obj = modello)

         1          2          3          4          5          6          7 
-0.3371620  0.1540936  0.4619923  0.2465971  0.0910624  0.2336206 -0.3426347 
         8 
-0.5075693 
```

residuals()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

  `type = "response" / "pearson"` tipo di residuo

- **Description:** residui
- **Formula:**

`type = "response"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$\sqrt{w_i}\, e_i \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> residuals(object = modello, type = "response")

         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227
```

- **Example 2:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> residuals(object = modello, type = "pearson")

         1          2          3          4          5          6          7
-0.3371620  0.1540936  0.4619923  0.2465971  0.0910624  0.2336206 -0.3426347
         8
-0.5075693
```

residuals.lm()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** residui

- **Formula:**

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> residuals.lm(object = modello)

         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227
```

residuals.default()

- **Package:** stats
- **Input:**

    object modello di regressione lineare pesata con $k - 1$ variabili esplicative ed n unità

- **Description:** residui
- **Formula:**

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> residuals.default(modello)

         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227
```

resid()

- **Package:** stats
- **Input:**

    object modello di regressione lineare pesata con $k - 1$ variabili esplicative ed n unità

- **Description:** residui
- **Formula:**

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> resid(object = modello)
```

```
         1          2          3          4          5          6          7
-0.9536382  0.4358424  1.3067117  0.6974820  0.2575634  0.6607787 -0.9691173
         8
-1.4356227
```

df.residual()

- **Package:** stats
- **Input:**

  `object` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** gradi di libertà della devianza residua
- **Formula:**

$$n - k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> df.residual(object = modello)
```

```
[1] 4
```

hatvalues()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> hatvalues(model = modello)
```

```
        1         2         3         4         5         6         7         8
0.7695906 0.4163361 0.3791092 0.3154744 0.7283511 0.5539241 0.4302463 0.4069682
```

hat()

- **Package:** stats
- **Input:**

  `x` matrice del modello

- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> X <- model.matrix(object = modello)
> hat(x = X)
```

```
[1] 0.7695906 0.4163361 0.3791092 0.3154744 0.7283511 0.5539241 0.4302463
[8] 0.4069682
```

dfbeta()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** dfbeta
- **Formula:**

$$\hat{\beta}_j - \hat{\beta}_{j\,(-i)} = w_i\, e_i\, (1-h_i)^{-1}\, (X^T\, W^{-1}\, X)^{-1}_j\, X_i^T \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> dfbeta(model = modello)
```

```
  (Intercept)          x1          x2          x3
1 -3.95445343  0.12758388  0.01022818  0.44042192
2  0.21929134  0.01923025 -0.12292616  0.08309302
3 -0.15505077  0.14594807 -0.39064531  0.32853997
4  0.10864633 -0.01436987  0.12965355 -0.11055404
5  0.06456839  0.14591697 -0.04391330 -0.06357315
6  0.27248353 -0.28472521  0.38742501 -0.16358023
7  0.36758841  0.18614884 -0.28071294  0.03129723
8  0.76981755 -0.23622669  0.37474061 -0.34716366
```

dfbetas()

- **Package:** stats
- **Input:**

  `model` modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** dfbetas
- **Formula:**

$$\frac{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}{s_{\hat{\beta}_j - \hat{\beta}_{j\,(-i)}}} = \frac{w_i\, e_i\, (1 - h_i)^{-1}\, (X^T\, W^{-1}\, X)_j^{-1}\, X_i^T}{s_{-i}\, \sqrt{(X^T\, W^{-1}\, X)_{j,\,j}^{-1}}} \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> dfbetas(model = modello)
```

```
  (Intercept)          x1          x2          x3
1 -3.70059595  0.43942641  0.02349647  1.44767218
2  0.13617748  0.04395152 -0.18739044  0.18124433
3 -0.12176106  0.42183052 -0.75307182  0.90623075
4  0.06957072 -0.03386642  0.20380513 -0.24865783
5  0.03984687  0.33142498 -0.06652573 -0.13780473
6  0.17845806 -0.68632053  0.62287782 -0.37630746
7  0.25592307  0.47699422 -0.47976587  0.07653668
8  0.66729165 -0.75363662  0.79740312 -1.05700791
```

vif()

- **Package:** car
- **Input:**

  `mod` modello di regressione lineare pesata con $k - 1$ variabili esplicative ed n unità
- **Description:** 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.
- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> vif(mod = modello)

      x1       x2       x3
4.133964 8.831535 3.758662
```

outlier.test()

- **Package:** car
- **Input:**

  `model` modello di regressione lineare pesata con $k - 1$ variabili esplicative ed n unità
- **Description:** test sugli *outliers*
- **Output:**

  `test` massimo residuo studentizzato assoluto, gradi di libertà, p-value
- **Formula:**

  `test`

$$t = \max_i(\,|\,rstudent_i\,|) \quad n - k - 1 \quad p\text{-value} = 2\,P(\,t_{n-k-1} \leq -|\,t\,|) \qquad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> outlier.test(model = modello)

max|rstudent| = 2.038485, degrees of freedom = 3,
unadjusted p = 0.1342423, Bonferroni p > 1

Observation: 1
```

```
> res <- outlier.test(model = modello)
> res$test

max|rstudent|             df  unadjusted p  Bonferroni p
    2.0384846      3.0000000      0.1342423            NA
```

influence.measures()

- **Package:** stats
- **Input:**

  model modello di regressione lineare pesata con $k-1$ variabili esplicative ed n unità

- **Description:** dfbetas, dffits, covratio, distanza di *Cook*, valori di leva
- **Output:**

  infmat misure di influenza di dimensione $n \times (k+4)$

  is.inf matrice di influenza con valori logici di dimensione $n \times (k+4)$

- **Formula:**

  infmat

$$DFBETAS_{ij} = \frac{w_i\, e_i\, (1-h_i)^{-1}\, (X^T\, W^{-1}\, X)_j^{-1}\, X_i^T}{s_{-i}\sqrt{(X^T\, W^{-1}\, X)_{j,\,j}^{-1}}} \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$$

$$DFFITS_i = rstudent_i \sqrt{\frac{h_i}{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$$

$$COVRATIO_i = (1-h_i)^{-1} \left(1 + \frac{rstudent_i^2 - 1}{n-k}\right)^{-k} \quad \forall i = 1, 2, \ldots, n$$

$$COOKD_i = \frac{h_i\, rstandard_i^2}{k\,(1-h_i)} \quad \forall\, i = 1, 2, \ldots, n$$

$$HAT_i = h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> k <- 4
> x1 <- c(1.1, 2.3, 4.5, 6.7, 8.9, 3.4, 5.6, 6.7)
> x2 <- c(1.2, 3.4, 5.6, 7.5, 7.5, 6.7, 8.6, 7.6)
> x3 <- c(1.4, 5.6, 7.56, 6, 5.4, 6.6, 8.7, 8.7)
> y <- c(1.5, 6.4, 9.6, 8.8, 8.86, 7.8, 8.6, 8.6)
> n <- 8
> modello <- lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n,
+     n))
> res <- influence.measures(model = modello)
> res

Influence measures of
         lm(formula = y ~ x1 + x2 + x3, weights = rep(1/n, n)) :

   dfb.1_  dfb.x1  dfb.x2  dfb.x3  dffit  cov.r cook.d   hat inf
1 -3.7006  0.4394  0.0235  1.4477 -3.726  0.424 1.9397 0.770   *
2  0.1362  0.0440 -0.1874  0.1812  0.328  4.450 0.0342 0.416   *
3 -0.1218  0.4218 -0.7531  0.9062  1.116  0.640 0.2471 0.379
4  0.0696 -0.0339  0.2038 -0.2487  0.402  2.968 0.0482 0.315
5  0.0398  0.3314 -0.0665 -0.1378  0.548 10.050 0.0963 0.728   *
6  0.1785 -0.6863  0.6229 -0.3763  0.792  3.804 0.1788 0.554
7  0.2559  0.4770 -0.4798  0.0765 -0.852  1.826 0.1832 0.430
8  0.6673 -0.7536  0.7974 -1.0570 -1.468  0.304 0.3509 0.407   *

> res$infmat
```

```
       dfb.1_      dfb.x1      dfb.x2      dfb.x3      dffit      cov.r
1 -3.70059595  0.43942641  0.02349647  1.44767218 -3.7255223  0.4238374
2  0.13617748  0.04395152 -0.18739044  0.18124433  0.3280660  4.4498753
3 -0.12176106  0.42183052 -0.75307182  0.90623075  1.1157578  0.6395729
4  0.06957072 -0.03386642  0.20380513 -0.24865783  0.4018144  2.9682483
5  0.03984687  0.33142498 -0.06652573 -0.13780473  0.5475321 10.0502975
6  0.17845806 -0.68632053  0.62287782 -0.37630746  0.7916935  3.8036903
7  0.25592307  0.47699422 -0.47976587  0.07653668 -0.8516950  1.8260516
8  0.66729165 -0.75363662  0.79740312 -1.05700791 -1.4677742  0.3038647
      cook.d       hat
1 1.93972080 0.7695906
2 0.03415783 0.4163361
3 0.24706215 0.3791092
4 0.04819074 0.3154744
5 0.09633983 0.7283511
6 0.17883712 0.5539241
7 0.18315058 0.4302463
8 0.35091186 0.4069682
```

```
> res$is.inf
```

```
  dfb.1_ dfb.x1 dfb.x2 dfb.x3 dffit cov.r cook.d   hat
1   TRUE  FALSE  FALSE   TRUE  TRUE FALSE   TRUE FALSE
2  FALSE  FALSE  FALSE  FALSE FALSE  TRUE  FALSE FALSE
3  FALSE  FALSE  FALSE  FALSE FALSE FALSE  FALSE FALSE
4  FALSE  FALSE  FALSE  FALSE FALSE FALSE  FALSE FALSE
5  FALSE  FALSE  FALSE  FALSE FALSE  TRUE  FALSE FALSE
6  FALSE  FALSE  FALSE  FALSE FALSE FALSE  FALSE FALSE
7  FALSE  FALSE  FALSE  FALSE FALSE FALSE  FALSE FALSE
8  FALSE  FALSE  FALSE   TRUE FALSE FALSE  FALSE FALSE
```

- **Note 1:** Il caso i-esimo è influente se $|DFBETAS_{ij}| > 1 \quad \forall i = 1, 2, \ldots, n \quad \forall j = 1, 2, \ldots, k$
- **Note 2:** Il caso i-esimo è influente se $|DFFITS_i| > 3\sqrt{k\,/\,(n-k)} \quad \forall i = 1, 2, \ldots, n$
- **Note 3:** Il caso i-esimo è influente se $|1 - COVRATIO_i| > 3\,k\,/\,(n-k) \quad \forall i = 1, 2, \ldots, n$
- **Note 4:** Il caso i-esimo è influente se $P\,(F_{k,n-k} \geq COOKD_i) > 0.5 \quad \forall i = 1, 2, \ldots, n$
- **Note 5:** Il caso i-esimo è influente se $HAT_i > 3\,k\,/\,n \quad \forall i = 1, 2, \ldots, n$
- **Note 6:** I casi influenti rispetto ad almeno una tra queste misure sono marcati con un asterisco. Corrispondentemente la stessa riga della matrice `is.inf` riporterà almeno un simbolo `TRUE`.

Parte V

Modelli Lineari Generalizzati

Capitolo 17

Regressione Logit

17.1 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 : $X_i = (1, x_{i1}, x_{i2}, \ldots, x_{ik-1}) \quad \forall\, i = 1, 2, \ldots, n$
- vettore numerico positivo dei pesi IWLS: $w = (w_1, w_2, \ldots, w_n)$
- 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 $n \times n$: $H = X\,(X^T\, W^{-1}\, X)^{-1}\, X^T\, W^{-1}$
- valori di leva: $h_i = H_{i,i} \quad \forall\, i = 1, 2, \ldots, n$
- distanza di *Cook*: $cd_i = \left(e_i^P\right)^2 \frac{h_i}{k\,(1-h_i)^2} \quad \forall\, i = 1, 2, \ldots, n$
- stime IWLS: $\hat{\beta} = \left(\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_k\right)^T$
- standard error delle stime IWLS: $s_{\hat{\beta}} = \sqrt{\text{diag}((X^T\, W^{-1}\, X)^{-1})}$
- z-values delle stime IWLS: $z_{\hat{\beta}} = \hat{\beta} \,/\, s_{\hat{\beta}}$
- correlazione delle stime IWLS: $r_{\hat{\beta}_i\, \hat{\beta}_j} = \frac{(X^T\, W^{-1}\, X)^{-1}_{i,\,j}}{s_{\hat{\beta}_i}\, s_{\hat{\beta}_j}} \quad \forall\, i, j = 1, 2, \ldots, k$
- residui di devianza: $e_i = \text{sign}\,(y_i - \hat{y}_i)\, \sqrt{2\left[y_i\, \log\left(\frac{y_i}{\hat{y}_i} + C_{i1}\right) + (n_i - y_i)\, \log\left(\frac{n_i - y_i}{n_i - \hat{y}_i} + C_{i2}\right)\right]}$

  $\forall\, i = 1, 2, \ldots, n$ dove $C_{i1} = 0.5\,(1 - \text{sign}(y_i))\,/\,\hat{y}_i$ e $C_{i2} = 0.5\,(1 - \text{sign}(n_i - y_i))\,/\,(n_i - \hat{y}_i)$
- residui standard: $rstandard_i = e_i \,/\, \sqrt{1 - h_i} \quad \forall\, i = 1, 2, \ldots, n$
- residui studentizzati: $rstudent_i = \text{sign}\,(y_i - \hat{y}_i)\, \sqrt{e_i^2 + h_i\, \left(e_i^P\right)^2 / (1 - h_i)} \quad \forall\, i = 1, 2, \ldots, n$
- residui di *Pearson*: $e_i^P = \frac{y_i - n_i\, \hat{\pi}_i}{\sqrt{n_i\, \hat{\pi}_i\, (1 - \hat{\pi}_i)}} \quad \forall i = 1, 2, \ldots, n$
- residui di lavoro: $e_i^W = \frac{y_i - n_i\, \hat{\pi}_i}{n_i\, \hat{\pi}_i\, (1 - \hat{\pi}_i)} \quad \forall i = 1, 2, \ldots, n$
- residui di riposta: $e_i^R = y_i \,/\, n_i - \hat{\pi}_i \quad \forall i = 1, 2, \ldots, n$
- log-verosimiglianza binomiale: $\hat{\ell} = \sum_{i=1}^n \left[\log\binom{n_i}{y_i} + y_i\, \log\left(\frac{\hat{y}_i}{n_i}\right) + (n_i - y_i)\, \log\left(1 - \frac{\hat{y}_i}{n_i}\right)\right]$
- valori adattati: $\hat{\pi}_i = \frac{\exp\left(X_i\, \hat{\beta}\right)}{1 + \exp\left(X_i\, \hat{\beta}\right)} \quad \forall\, i = 1, 2, \ldots, n$

- numero di successi attesi: $\hat{y}_i = n_i\, \hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$
- log-verosimiglianza binomiale modello saturo: $\hat{\ell}_{saturo} = \sum_{i=1}^{n} \left[\log \binom{n_i}{y_i} + y_i \log\left(\frac{y_i}{n_i}\right) + (n_i - y_i) \log\left(1 - \frac{y_i}{n_i}\right)\right]$
- devianza residua: $D = 2\left(\hat{\ell}_{saturo} - \hat{\ell}\right) = \sum_{i=1}^{n} e_i^2$
- gradi di libertà della devianza residua: $n - k$
- log-verosimiglianza binomiale modello nullo: $\hat{\ell}_{nullo} = \sum_{i=1}^{n} \left[\log \binom{n_i}{y_i} + y_i \log\left(\hat{\pi}\right) + (n_i - y_i) \log\left(1 - \hat{\pi}\right)\right]$
- valori adattati modello nullo: $\hat{\pi} = \sum_{j=1}^{n} y_j \,/\, \sum_{j=1}^{n} n_j \quad \forall\, i = 1, 2, \ldots, n$
- numero di successi attesi modello nullo: $\hat{y}_i = n_i\, \hat{\pi} \quad \forall\, i = 1, 2, \ldots, n$
- devianza residua modello nullo: $D_{nullo} = 2\left(\hat{\ell}_{saturo} - \hat{\ell}_{nullo}\right)$
- gradi di libertà della devianza residua modello nullo: $n - 1$
- stima IWLS intercetta modello nullo: $\hat{\beta}_{nullo} = \log\left(\frac{\hat{\pi}}{1-\hat{\pi}}\right)$

17.2 Stima

glm()

- **Package:** `stats`
- **Input:**

  `formula` modello di regressione logit con $k - 1$ variabili esplicative ed n unità

  `family = binomial(link="logit")` famiglia e link del modello

  `x = TRUE` matrice del modello

- **Description:** analisi di regressione logit
- **Output:**

  `coefficients` stime IWLS

  `residuals` residui di lavoro

  `fitted.values` valori adattati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` 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`

  $$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

  `residuals`

  $$e_i^W \quad \forall\, i = 1, 2, \ldots, n$$

  `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

$$X$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"),
+     x = TRUE)
> modello$coefficients

(Intercept)           x
 -21.226395    1.631968

> modello$residuals

          1           2           3           4           5           6
-1.00203763 -1.01042031 -1.01905988 -0.41336424 -0.48212701 -0.07089826
          7           8           9          10          11          12
 0.07938086  0.22704866 -0.13926878  0.33629857  0.25835047  0.17881393
         13          14          15          16          17          18
-0.22141017  0.01336452  0.26283804 -0.24965088 -0.36552096  0.33713195
         19          20          21          22          23          24
 0.19514514 -0.43506531 -0.25760272 -0.64783388 -0.44626460 -0.78405425
         25
 1.00057358

> modello$fitted.values
```

```
          1           2           3           4           5           6
0.002033490 0.010312851 0.018703394 0.027863526 0.041320994 0.060871141
          7           8           9          10          11          12
0.088814107 0.127838223 0.180610428 0.248949062 0.332647930 0.428434554
         13          14          15          16          17          18
0.529902047 0.628956590 0.718237396 0.793102235 0.852169542 0.896572801
         19          20          21          22          23          24
0.928753893 0.951463983 0.967190831 0.977939948 0.985221193 0.990123427
         25
0.999426746
```

```
> modello$rank
```

```
[1] 2
```

```
> modello$linear.predictors
```

```
         1          2          3          4          5          6          7
-6.1959664 -4.5639981 -3.9601698 -3.5521777 -3.1441856 -2.7361935 -2.3282014
         8          9         10         11         12         13         14
-1.9202093 -1.5122173 -1.1042252 -0.6962331 -0.2882410  0.1197511  0.5277432
        15         16         17         18         19         20         21
 0.9357353  1.3437274  1.7517194  2.1597115  2.5677036  2.9756957  3.3836878
        22         23         24         25
 3.7916799  4.1996720  4.6076640  7.4636087
```

```
> modello$deviance
```

```
[1] 26.70345
```

```
> modello$aic
```

```
[1] 114.7553
```

```
> modello$null.deviance
```

```
[1] 3693.884
```

```
> modello$weights
```

```
         1          2          3          4          5          6          7
 0.7630428  2.0413099  1.7068902  3.2504707  3.5652333  5.0306085  8.4972661
         8          9         10         11         12         13         14
12.3760338 14.7990471 17.3885402 22.1993347 26.4468672 24.6614810 24.7372446
        15         16         17         18         19         20         21
21.2491158 19.1986735 12.3457255  8.9948289  7.9404319  4.7104022  3.8714069
        22         23         24         25
 2.3946581  1.3686835  1.1148148  0.6010036
```

```
> modello$prior.weights
```

```
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
 376  200   93  120   90   88  105  111  100   93  100  108   99  106  105  117
  17   18   19   20   21   22   23   24   25
  98   97  120  102  122  111   94  114 1049
```

```
> modello$df.residual
```

```
[1] 23
```

```
> modello$df.null

[1] 24

> modello$y

         1          2          3          4          5          6          7
0.00000000 0.00000000 0.00000000 0.01666667 0.02222222 0.05681818 0.09523810
         8          9         10         11         12         13         14
0.15315315 0.16000000 0.31182796 0.39000000 0.47222222 0.47474747 0.63207547
        15         16         17         18         19         20         21
0.77142857 0.75213675 0.80612245 0.92783505 0.94166667 0.93137255 0.95901639
        22         23         24         25
0.96396396 0.97872340 0.98245614 1.00000000

> modello$x

   (Intercept)     x
1            1  9.21
2            1 10.21
3            1 10.58
4            1 10.83
5            1 11.08
6            1 11.33
7            1 11.58
8            1 11.83
9            1 12.08
10           1 12.33
11           1 12.58
12           1 12.83
13           1 13.08
14           1 13.33
15           1 13.58
16           1 13.83
17           1 14.08
18           1 14.33
19           1 14.58
20           1 14.83
21           1 15.08
22           1 15.33
23           1 15.58
24           1 15.83
25           1 17.58
attr(,"assign")
[1] 0 1
```

summary.glm()

- **Package:** stats
- **Input:**

  `object` modello di regressione logit con $k-1$ variabili esplicative ed n unità

  `correlation = TRUE` correlazione delle stime IWLS

- **Description:** 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 IWLS non scalata

`cov.scaled` matrice di covarianza delle stime IWLS scalata

`correlation` matrice di correlazione delle stime IWLS

- **Formula:**

`deviance`

$$D$$

`aic`

$$-2\,\hat{\ell} + 2\,k$$

`df.residual`

$$n - k$$

`null.deviance`

$$D_{nullo}$$

`df.null`

$$n - 1$$

`deviance.resid`

$$e_i \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

`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`

$$(X^T\,W^{-1}\,X)^{-1}$$

`correlation`

$$r_{\hat{\beta}_i\,\hat{\beta}_j} \quad \forall\, i, j \,=\, 1,\, 2,\, \ldots,\, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> res <- summary.glm(object = modello, correlation = TRUE)
> res$deviance

[1] 26.70345

> res$aic

[1] 114.7553

> res$df.residual

[1] 23
```

```
> res$null.deviance

[1] 3693.884

> res$df.null

[1] 24

> res$deviance.resid

         1          2          3          4          5          6          7
-1.2372312 -2.0363101 -1.8739732 -0.8043827 -0.9953320 -0.1607163  0.2289532
         8          9         10         11         12         13         14
 0.7780252 -0.5441548  1.3675388  1.2016944  0.9162826 -1.0982255  0.0665090
        15         16         17         18         19         20         21
 1.2375553 -1.0695134 -1.2358120  1.0633044  0.5665503 -0.8912577 -0.4883964
        22         23         24         25
-0.9195743 -0.4900070 -0.7461893  1.0968278

> res$coefficients

              Estimate Std. Error   z value      Pr(>|z|)
(Intercept) -21.226395 0.77068466 -27.54226 5.479038e-167
x             1.631968 0.05895308  27.68249 1.134448e-168

> res$cov.unscaled

            (Intercept)            x
(Intercept)  0.59395485 -0.045281754
x           -0.04528175  0.003475466

> res$cov.scaled

            (Intercept)            x
(Intercept)  0.59395485 -0.045281754
x           -0.04528175  0.003475466

> res$correlation

            (Intercept)         x
(Intercept)    1.000000 -0.996644
x             -0.996644  1.000000
```

glm.fit()

- **Package:** `stats`
- **Input:**

  `x` matrice del modello

  `y` proporzione di successi

  `weights` numero di prove

  `family = binomial(link="logit")` famiglia e link del modello

- **Description:** analisi di regressione logit
- **Output:**

  `coefficients` stime IWLS

`residuals` residui di lavoro
`fitted.values` valori adattati
`rank` rango della matrice del modello
`linear.predictors` predittori lineari
`deviance` 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`
$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

`residuals`
$$e_i^W \quad \forall\, i = 1, 2, \ldots, n$$

`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`
$$y_i \,/\, n_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> X <- model.matrix(object = modello)
> res <- glm.fit(x = X, y/Total, weights = Total, family = binomial(link = "logit"))
> res$coefficients
```

```
(Intercept)           x
 -21.226395    1.631968

> res$residuals

 [1] -1.00203763 -1.01042031 -1.01905988 -0.41336424 -0.48212701 -0.07089826
 [7]  0.07938086  0.22704866 -0.13926878  0.33629857  0.25835047  0.17881393
[13] -0.22141017  0.01336452  0.26283804 -0.24965088 -0.36552096  0.33713195
[19]  0.19514514 -0.43506531 -0.25760272 -0.64783388 -0.44626460 -0.78405425
[25]  1.00057358

> res$fitted.values

 [1] 0.002033490 0.010312851 0.018703394 0.027863526 0.041320994 0.060871141
 [7] 0.088814107 0.127838223 0.180610428 0.248949062 0.332647930 0.428434554
[13] 0.529902047 0.628956590 0.718237396 0.793102235 0.852169542 0.896572801
[19] 0.928753893 0.951463983 0.967190831 0.977939948 0.985221193 0.990123427
[25] 0.999426746

> res$rank

[1] 2

> res$linear.predictors

 [1] -6.1959664 -4.5639981 -3.9601698 -3.5521777 -3.1441856 -2.7361935
 [7] -2.3282014 -1.9202093 -1.5122173 -1.1042252 -0.6962331 -0.2882410
[13]  0.1197511  0.5277432  0.9357353  1.3437274  1.7517194  2.1597115
[19]  2.5677036  2.9756957  3.3836878  3.7916799  4.1996720  4.6076640
[25]  7.4636087

> res$deviance

[1] 26.70345

> res$aic

[1] 114.7553

> res$null.deviance

[1] 3693.884

> res$weights

 [1]  0.7630428  2.0413099  1.7068902  3.2504707  3.5652333  5.0306085
 [7]  8.4972661 12.3760338 14.7990471 17.3885402 22.1993347 26.4468672
[13] 24.6614810 24.7372446 21.2491158 19.1986735 12.3457255  8.9948289
[19]  7.9404319  4.7104022  3.8714069  2.3946581  1.3686835  1.1148148
[25]  0.6010036

> res$prior.weights

 [1]  376  200   93  120   90   88  105  111  100   93  100  108   99  106  105
[16]  117   98   97  120  102  122  111   94  114 1049

> res$df.residual

[1] 23
```

```
> res$df.null
```

```
[1] 24
```

```
> res$y
```

```
 [1] 0.00000000 0.00000000 0.00000000 0.01666667 0.02222222 0.05681818
 [7] 0.09523810 0.15315315 0.16000000 0.31182796 0.39000000 0.47222222
[13] 0.47474747 0.63207547 0.77142857 0.75213675 0.80612245 0.92783505
[19] 0.94166667 0.93137255 0.95901639 0.96396396 0.97872340 0.98245614
[25] 1.00000000
```

vcov()

- **Package:** stats
- **Input:**

  `object` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** matrice di covarianza delle stime IWLS
- **Formula:**

$$(X^T W^{-1} X)^{-1}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> vcov(object = modello)
```

```
            (Intercept)            x
(Intercept)  0.59395485 -0.045281754
x           -0.04528175  0.003475466
```

coef()

- **Package:** stats
- **Input:**

  `object` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \dots, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> coef(object = modello)

(Intercept)           x
 -21.226395    1.631968
```

coefficients()

- **Package:** stats
- **Input:**

  object modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> coefficients(object = modello)

(Intercept)           x
 -21.226395    1.631968
```

predict.glm()

- **Package:** stats
- **Input:**

  object modello di regressione logit con $k-1$ variabili esplicative ed n unità

  newdata il valore di x_0

  se.fit = TRUE standard error delle stime

- **Description:** previsione
- **Output:**

  fit valore previsto

  se.fit standard error delle stime

- **Formula:**

fit

$$x_0^T\,\hat{\beta}$$

se.fit

$$\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> res <- predict.glm(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit

        1
-19.10484

> res$se.fit

[1] 0.6943312
```

predict()

- **Package:** stats
- **Input:**

  object modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

  newdata il valore di x_0

  se.fit = TRUE standard error delle stime
- **Description:** previsione
- **Output:**

  fit valore previsto

  se.fit standard error delle stime
- **Formula:**

fit

$$x_0^T\,\hat{\beta}$$

se.fit

$$\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> res <- predict.glm(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit
```

```
        1
-19.10484
```

```
> res$se.fit
```

```
[1] 0.6943312
```

fitted()

- **Package:** stats
- **Input:**

  `object` modello di regressione logit con $k - 1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> fitted(object = modello)
```

```
          1           2           3           4           5           6
0.002033490 0.010312851 0.018703394 0.027863526 0.041320994 0.060871141
          7           8           9          10          11          12
0.088814107 0.127838223 0.180610428 0.248949062 0.332647930 0.428434554
         13          14          15          16          17          18
0.529902047 0.628956590 0.718237396 0.793102235 0.852169542 0.896572801
         19          20          21          22          23          24
0.928753893 0.951463983 0.967190831 0.977939948 0.985221193 0.990123427
         25
0.999426746
```

fitted.values()

- **Package:** stats
- **Input:**

  `object` modello di regressione logit con $k - 1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> fitted.values(object = modello)

          1           2           3           4           5           6
0.002033490 0.010312851 0.018703394 0.027863526 0.041320994 0.060871141
          7           8           9          10          11          12
0.088814107 0.127838223 0.180610428 0.248949062 0.332647930 0.428434554
         13          14          15          16          17          18
0.529902047 0.628956590 0.718237396 0.793102235 0.852169542 0.896572801
         19          20          21          22          23          24
0.928753893 0.951463983 0.967190831 0.977939948 0.985221193 0.990123427
         25
0.999426746
```

cov2cor()

- **Package:** stats
- **Input:**

  V matrice di covarianza delle stime IWLS di dimensione $k \times k$

- **Description:** converte la matrice di covarianza nella matrice di correlazione
- **Formula:**

$$r_{\hat{\beta}_i \, \hat{\beta}_j} \quad \forall\, i, j \,=\, 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> V <- vcov(object = modello)
> cov2cor(V)

            (Intercept)         x
(Intercept)    1.000000 -0.996644
x             -0.996644  1.000000
```

17.3 Adattamento

logLik()

- **Package:** stats
- **Input:**

`object` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** log-verosimiglianza binomiale
- **Formula:**

$$\hat{\ell}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> logLik(object = modello)

'log Lik.' -55.37763 (df=2)
```

AIC()

- **Package:** stats
- **Input:**

`object` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> AIC(object = modello)

[1] 114.7553
```

durbin.watson()

- **Package:** car
- **Input:**

`model` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

`dw` valore empirico della statistica *D–W*

- **Formula:**

  dw

$$\sum_{i=2}^{n} (e_i - e_{i-1})^2 \,/\, D$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> durbin.watson(model = modello)
```

```
 lag Autocorrelation D-W Statistic p-value
   1       0.3440895      1.209446   0.034
 Alternative hypothesis: rho != 0
```

```
> res <- durbin.watson(model = modello)
> res$dw
```

```
[1] 1.209446
```

extractAIC()

- **Package:** stats
- **Input:**

  `fit` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** numero di parametri del modello ed indice *AIC* generalizzato
- **Formula:**

$$k \qquad -2\,\hat{\ell} + 2\,k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> extractAIC(fit = modello)
```

```
[1]   2.0000 114.7553
```

deviance()

- **Package:** stats
- **Input:**

  `object` modello di regressione logit con $k-1$ variabili esplicative ed n unità
- **Description:** devianza residua
- **Formula:**

$$D$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> deviance(object = modello)

[1] 26.70345
```

anova()

- **Package:** stats
- **Input:**

  `nullo` modello nullo di regressione logit con n unità

  `modello` modello di regressione logit con $k-1$ variabili esplicative con n unità

  `test = "Chisq"`
- **Description:** 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`

$$df = k-1$$

  `Deviance`

$$c = D_{nullo} - D$$

  `P(>|Chi|)`

$$P(\chi^2_{df} \geq c)$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> nullo <- glm(formula = cbind(y, Total - y) ~ 1, family = binomial(link = "logit"))
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> anova(nullo, modello, test = "Chisq")

Analysis of Deviance Table

Model 1: cbind(y, Total - y) ~ 1
Model 2: cbind(y, Total - y) ~ x
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1        24     3693.9
2        23       26.7  1   3667.2       0.0

> res <- anova(nullo, modello, test = "Chisq")
> res$"Resid. Df"

[1] 24 23

> res$"Resid. Dev"

[1] 3693.88357   26.70345

> res$Df

[1] NA  1

> res$Deviance

[1]      NA 3667.18

> res$"P(>|Chi|)"

[1] NA  0
```

drop1()

- **Package:** stats
- **Input:**

  `object` modello di regressione logit con $k-1$ variabili esplicative ed n unità

  `test = "Chisq"`
- **Description:** submodels
- **Output:**

  `Df` differenza tra gradi di libertà

  `Deviance` differenza tra devianze residue

  `AIC` indice *AIC*

  `LRT` valore empirico della statistica χ^2

  `Pr(Chi)` p-value

- **Formula:**

  `Df`

$$\underbrace{1,\,1,\,\dots,\,1}_{k-1 \text{ volte}}$$

  `Deviance`

$$D,\,D_{-x_j} \quad \forall\, j \,=\, 1,\,2,\,\dots,\,k-1$$

  dove D_{-x_j} rappresenta la devianza residua del modello eliminata la variabile esplicativa x_j.

  `AIC`

$$-2\,\hat{\ell} + 2\,k,\; -2\,\hat{\ell}_{-x_j} + 2\,(k-1) \quad \forall\, j \,=\, 1,\,2,\,\dots,\,k-1$$

  dove $\hat{\ell}_{-x_j}$ rappresenta la log-verosimiglianza binomiale del modello eliminata la variabile esplicativa x_j.

  `LRT`

$$c_j \,=\, D_{-x_j} - D \quad \forall\, j \,=\, 1,\,2,\,\dots,\,k-1$$

  `Pr(Chi)`

$$P(\chi^2_1 \geq c_j) \quad \forall\, j \,=\, 1,\,2,\,\dots,\,k-1$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> drop1(object = modello, test = "Chisq")

Single term deletions

Model:
cbind(y, Total - y) ~ x
       Df Deviance    AIC    LRT   Pr(Chi)
<none>        26.7  114.8
x       1   3693.9 3779.9 3667.2 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> res <- drop1(object = modello, test = "Chisq")
> res$Df

[1] NA  1

> res$Deviance

[1]   26.70345 3693.88357

> res$AIC

[1]  114.7553 3779.9354

> res$LRT

[1]      NA 3667.18

> res$"Pr(Chi)"

[1] NA  0
```

add1()

- **Package:** stats
- **Input:**

  `object` modello nullo di regressione logit

  `scope` modello di regressione logit con $k-1$ variabili esplicative ed n unità

  `test = "Chisq"`

- **Description:** submodels
- **Output:**

  `Df` differenza tra gradi di libertà

  `Deviance` differenza tra devianze residue

  `AIC` indice AIC

  `LRT` valore empirico della statistica χ^2

  `Pr(Chi)` p-value

- **Formula:**

  `Df`

  $$\underbrace{1,\, 1,\, \ldots,\, 1}_{k-1 \text{ volte}}$$

  `Deviance`

  $$D_{nullo},\, D_{x_j} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  dove D_{x_j} rappresenta la devianza residua del modello con la sola variabile esplicativa x_j.

  `AIC`

  $$-2\,\hat{\ell}_{nullo} + 2,\, -2\,\hat{\ell}_{x_j} + 4 \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  dove $\hat{\ell}_{x_j}$ rappresenta la log-verosimiglianza binomiale del modello con la sola variabile esplicativa x_j.

  `LRT`

  $$c_j \,=\, D_{nullo} - D_{x_j} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  `Pr(Chi)`

  $$P(\chi^2_1 \geq c_j) \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> nullo <- glm(formula = cbind(y, Total - y) ~ 1, family = binomial(link = "logit"))
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> add1(object = nullo, scope = modello, test = "Chisq")

Single term additions

Model:
cbind(y, Total - y) ~ 1
       Df Deviance    AIC    LRT   Pr(Chi)
<none>      3693.9 3779.9
x       1     26.7  114.8 3667.2 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> res <- add1(object = nullo, scope = modello, test = "Chisq")
> res$Df

[1] NA  1

> res$Deviance

[1] 3693.88357   26.70345

> res$AIC

[1] 3779.9354  114.7553

> res$LRT

[1]      NA 3667.18

> res$"Pr(Chi)"

[1] NA  0
```

17.4 Diagnostica

rstandard()

- **Package:** stats
- **Input:**

  model modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> rstandard(model = modello)

          1           2           3           4           5           6
-1.26387269 -2.10534096 -1.91498313 -0.83301527 -1.02729335 -0.16669886
          7           8           9          10          11          12
 0.24077974  0.82521025 -0.57526008  1.44049872  1.26945542  0.97065728
         13          14          15          16          17          18
-1.15658902  0.07035119  1.30959757 -1.13960327 -1.30015928  1.11385953
         19          20          21          22          23          24
 0.59653144 -0.92511157 -0.50699153 -0.94525426 -0.49917710 -0.75953595
         25
 1.12275650
```

rstandard.glm()

- **Package:** stats
- **Input:**

  `model` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> rstandard.glm(model = modello)

          1           2           3           4           5           6
-1.26387269 -2.10534096 -1.91498313 -0.83301527 -1.02729335 -0.16669886
          7           8           9          10          11          12
 0.24077974  0.82521025 -0.57526008  1.44049872  1.26945542  0.97065728
         13          14          15          16          17          18
-1.15658902  0.07035119  1.30959757 -1.13960327 -1.30015928  1.11385953
         19          20          21          22          23          24
 0.59653144 -0.92511157 -0.50699153 -0.94525426 -0.49917710 -0.75953595
         25
 1.12275650
```

rstudent()

- **Package:** stats
- **Input:**

  `model` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> rstudent(model = modello)
```

```
          1           2           3           4           5           6
-1.25063645 -2.07129265 -1.89478391 -0.82902073 -1.02213647 -0.16657527
          7           8           9          10          11          12
 0.24102704  0.82768067 -0.57433275  1.44416053  1.27117259  0.97103803
         13          14          15          16          17          18
-1.15672425  0.07034687  1.30668616 -1.14272936 -1.30517189  1.10911742
         19          20          21          22          23          24
 0.59483577 -0.92917154 -0.50839548 -0.95001692 -0.50040422 -0.76258344
         25
 1.10987159
```

rstudent.glm()

- **Package:** stats
- **Input:**

  `model` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i \, = \, 1, \, 2, \, \ldots, \, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> rstudent.glm(model = modello)

          1           2           3           4           5           6
-1.25063645 -2.07129265 -1.89478391 -0.82902073 -1.02213647 -0.16657527
          7           8           9          10          11          12
 0.24102704  0.82768067 -0.57433275  1.44416053  1.27117259  0.97103803
         13          14          15          16          17          18
-1.15672425  0.07034687  1.30668616 -1.14272936 -1.30517189  1.10911742
         19          20          21          22          23          24
 0.59483577 -0.92917154 -0.50839548 -0.95001692 -0.50040422 -0.76258344
         25
 1.10987159
```

residuals.default()

- **Package:** stats
- **Input:**

  `object` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** residui di lavoro
- **Formula:**

$$e_i^W \quad \forall i \, = \, 1, \, 2, \, \ldots, \, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> residuals.default(object = modello)

          1           2           3           4           5           6
-1.00203763 -1.01042031 -1.01905988 -0.41336424 -0.48212701 -0.07089826
          7           8           9          10          11          12
 0.07938086  0.22704866 -0.13926878  0.33629857  0.25835047  0.17881393
         13          14          15          16          17          18
-0.22141017  0.01336452  0.26283804 -0.24965088 -0.36552096  0.33713195
         19          20          21          22          23          24
 0.19514514 -0.43506531 -0.25760272 -0.64783388 -0.44626460 -0.78405425
         25
 1.00057358
```

residuals()

- **Package:** stats
- **Input:**

  `object` modello di regressione logit con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo
- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> residuals(object = modello, type = "deviance")
```

```
        1          2          3          4          5          6          7
-1.2372312 -2.0363101 -1.8739732 -0.8043827 -0.9953320 -0.1607163  0.2289532
         8          9         10         11         12         13         14
 0.7780252 -0.5441548  1.3675388  1.2016944  0.9162826 -1.0982255  0.0665090
        15         16         17         18         19         20         21
 1.2375553 -1.0695134 -1.2358120  1.0633044  0.5665503 -0.8912577 -0.4883964
        22         23         24         25
-0.9195743 -0.4900070 -0.7461893  1.0968278
```

- **Example 2:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> residuals(object = modello, type = "pearson")

          1           2           3           4           5           6
-0.87529996 -1.44362837 -1.33137848 -0.74525548 -0.91034225 -0.15901761
          7           8           9          10          11          12
 0.23139551  0.79874716 -0.53576012  1.40235004  1.21724831  0.91957777
         13          14          15          16          17          18
-1.09953015  0.06647053  1.21159801 -1.09387707 -1.28431127  1.01110426
         19          20          21          22          23          24
 0.54989436 -0.94424085 -0.50685539 -1.00250029 -0.52208706 -0.82783987
         25
 0.77568558
```

- **Example 3:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> residuals(object = modello, type = "working")

          1           2           3           4           5           6
-1.00203763 -1.01042031 -1.01905988 -0.41336424 -0.48212701 -0.07089826
          7           8           9          10          11          12
 0.07938086  0.22704866 -0.13926878  0.33629857  0.25835047  0.17881393
         13          14          15          16          17          18
-0.22141017  0.01336452  0.26283804 -0.24965088 -0.36552096  0.33713195
         19          20          21          22          23          24
 0.19514514 -0.43506531 -0.25760272 -0.64783388 -0.44626460 -0.78405425
         25
 1.00057358
```

- **Example 4:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
```

```
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> residuals(object = modello, type = "response")

            1             2             3             4             5
-0.0020334895 -0.0103128513 -0.0187033936 -0.0111968589 -0.0190987716
            6             7             8             9            10
-0.0040529588  0.0064239884  0.0253149298 -0.0206104280  0.0628788951
           11            12            13            14            15
 0.0573520700  0.0437876678 -0.0551545725  0.0031188816  0.0531911753
           16            17            18            19            20
-0.0409654825 -0.0460470931  0.0312622502  0.0129127734 -0.0200914343
           21            22            23            24            25
-0.0081744371 -0.0139759836 -0.0064977884 -0.0076672869  0.0005732538
```

residuals.glm()

- **Package:** stats
- **Input:**

  `object` modello di regressione logit con $k - 1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo

- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> residuals.glm(object = modello, type = "deviance")
```

```
          1          2          3          4          5          6          7
-1.2372312 -2.0363101 -1.8739732 -0.8043827 -0.9953320 -0.1607163  0.2289532
         8          9         10         11         12         13         14
 0.7780252 -0.5441548  1.3675388  1.2016944  0.9162826 -1.0982255  0.0665090
        15         16         17         18         19         20         21
 1.2375553 -1.0695134 -1.2358120  1.0633044  0.5665503 -0.8912577 -0.4883964
        22         23         24         25
-0.9195743 -0.4900070 -0.7461893  1.0968278
```

- **Example 2:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> residuals.glm(object = modello, type = "pearson")

          1           2           3           4           5           6
-0.87529996 -1.44362837 -1.33137848 -0.74525548 -0.91034225 -0.15901761
          7           8           9          10          11          12
 0.23139551  0.79874716 -0.53576012  1.40235004  1.21724831  0.91957777
         13          14          15          16          17          18
-1.09953015  0.06647053  1.21159801 -1.09387707 -1.28431127  1.01110426
         19          20          21          22          23          24
 0.54989436 -0.94424085 -0.50685539 -1.00250029 -0.52208706 -0.82783987
         25
 0.77568558
```

- **Example 3:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> residuals.glm(object = modello, type = "working")

          1           2           3           4           5           6
-1.00203763 -1.01042031 -1.01905988 -0.41336424 -0.48212701 -0.07089826
          7           8           9          10          11          12
 0.07938086  0.22704866 -0.13926878  0.33629857  0.25835047  0.17881393
         13          14          15          16          17          18
-0.22141017  0.01336452  0.26283804 -0.24965088 -0.36552096  0.33713195
         19          20          21          22          23          24
 0.19514514 -0.43506531 -0.25760272 -0.64783388 -0.44626460 -0.78405425
         25
 1.00057358
```

- **Example 4:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
```

```
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> residuals.glm(object = modello, type = "response")
```

```
            1             2             3             4             5
-0.0020334895 -0.0103128513 -0.0187033936 -0.0111968589 -0.0190987716
            6             7             8             9            10
-0.0040529588  0.0064239884  0.0253149298 -0.0206104280  0.0628788951
           11            12            13            14            15
 0.0573520700  0.0437876678 -0.0551545725  0.0031188816  0.0531911753
           16            17            18            19            20
-0.0409654825 -0.0460470931  0.0312622502  0.0129127734 -0.0200914343
           21            22            23            24            25
-0.0081744371 -0.0139759836 -0.0064977884 -0.0076672869  0.0005732538
```

resid()

- **Package:** stats
- **Input:**

  object modello di regressione logit con $k-1$ variabili esplicative ed n unità

  type = "deviance" / "pearson" / "working" / "response" tipo di residuo

- **Description:** residui
- **Formula:**

type = "deviance"

$$e_i \quad \forall i = 1, 2, \ldots, n$$

type = "pearson"

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

type = "working"

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

type = "response"

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> resid(object = modello, type = "deviance")
```

```
        1          2          3          4          5          6          7
-1.2372312 -2.0363101 -1.8739732 -0.8043827 -0.9953320 -0.1607163  0.2289532
        8          9         10         11         12         13         14
 0.7780252 -0.5441548  1.3675388  1.2016944  0.9162826 -1.0982255  0.0665090
       15         16         17         18         19         20         21
 1.2375553 -1.0695134 -1.2358120  1.0633044  0.5665503 -0.8912577 -0.4883964
       22         23         24         25
-0.9195743 -0.4900070 -0.7461893  1.0968278
```

- **Example 2:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> resid(object = modello, type = "pearson")

          1           2           3           4           5           6
-0.87529996 -1.44362837 -1.33137848 -0.74525548 -0.91034225 -0.15901761
          7           8           9          10          11          12
 0.23139551  0.79874716 -0.53576012  1.40235004  1.21724831  0.91957777
         13          14          15          16          17          18
-1.09953015  0.06647053  1.21159801 -1.09387707 -1.28431127  1.01110426
         19          20          21          22          23          24
 0.54989436 -0.94424085 -0.50685539 -1.00250029 -0.52208706 -0.82783987
         25
 0.77568558
```

- **Example 3:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> resid(object = modello, type = "working")

          1           2           3           4           5           6
-1.00203763 -1.01042031 -1.01905988 -0.41336424 -0.48212701 -0.07089826
          7           8           9          10          11          12
 0.07938086  0.22704866 -0.13926878  0.33629857  0.25835047  0.17881393
         13          14          15          16          17          18
-0.22141017  0.01336452  0.26283804 -0.24965088 -0.36552096  0.33713195
         19          20          21          22          23          24
 0.19514514 -0.43506531 -0.25760272 -0.64783388 -0.44626460 -0.78405425
         25
 1.00057358
```

- **Example 4:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
```

```
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> resid(object = modello, type = "response")

            1             2             3             4             5
-0.0020334895 -0.0103128513 -0.0187033936 -0.0111968589 -0.0190987716
            6             7             8             9            10
-0.0040529588  0.0064239884  0.0253149298 -0.0206104280  0.0628788951
           11            12            13            14            15
 0.0573520700  0.0437876678 -0.0551545725  0.0031188816  0.0531911753
           16            17            18            19            20
-0.0409654825 -0.0460470931  0.0312622502  0.0129127734 -0.0200914343
           21            22            23            24            25
-0.0081744371 -0.0139759836 -0.0064977884 -0.0076672869  0.0005732538
```

weighted.residuals()

- **Package:** stats
- **Input:**

  `obj` modello di regressione logit con $k - 1$ variabili esplicative ed n unità

- **Description:** residui pesati
- **Formula:**

$$e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> weighted.residuals(obj = modello)

         1          2          3          4          5          6          7
-1.2372312 -2.0363101 -1.8739732 -0.8043827 -0.9953320 -0.1607163  0.2289532
         8          9         10         11         12         13         14
 0.7780252 -0.5441548  1.3675388  1.2016944  0.9162826 -1.0982255  0.0665090
        15         16         17         18         19         20         21
 1.2375553 -1.0695134 -1.2358120  1.0633044  0.5665503 -0.8912577 -0.4883964
        22         23         24         25
-0.9195743 -0.4900070 -0.7461893  1.0968278
```

weights()

- **Package:** stats
- **Input:**

  `object` modello di regressione logit con $k - 1$ variabili esplicative ed n unità

- **Description:** pesi iniziali

- **Formula:**

$$n_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> weights(object = modello)
```

```
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
 376  200   93  120   90   88  105  111  100   93  100  108   99  106  105  117
  17   18   19   20   21   22   23   24   25
  98   97  120  102  122  111   94  114 1049
```

df.residual()

- **Package:** stats
- **Input:**

  `object` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** gradi di libertà della devianza residua
- **Formula:**

$$n - k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> df.residual(object = modello)
```

```
[1] 23
```

hatvalues()

- **Package:** stats
- **Input:**

  `model` modello di regressione logit con $k-1$ variabili esplicative ed n unità

- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> hatvalues(model = modello)
```

```
         1          2          3          4          5          6          7
0.04171418 0.06450180 0.04237196 0.06756306 0.06125644 0.07048903 0.09582267
         8          9         10         11         12         13         14
0.11108936 0.10521957 0.09873284 0.10390681 0.10889885 0.09837709 0.10624609
        15         16         17         18         19         20         21
0.10699575 0.11922484 0.09653421 0.08871474 0.09799217 0.07184963 0.07200939
        22         23         24         25
0.05359644 0.03640349 0.03483536 0.04565424
```

cooks.distance()

- **Package:** stats
- **Input:**

  `model` modello di regressione logit con $k - 1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> cooks.distance(model = modello)
```

```
           1            2            3            4            5            6
0.0174011270 0.0768009809 0.0409503781 0.0215799628 0.0288029684 0.0010315088
           7            8            9           10           11           12
0.0031379129 0.0448481919 0.0188614178 0.1195191319 0.0958663105 0.0579850735
          13           14           15           16           17           18
0.0731523657 0.0002938362 0.0984796718 0.0919482890 0.0975367746 0.0546070811
          19           20           21           22           23           24
0.0182095530 0.0371812046 0.0107408856 0.0300692243 0.0053432866 0.0128138673
          25
0.0150803356
```

cookd()

- **Package:** car
- **Input:**

  model modello di regressione logit con $k - 1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> cookd(model = modello)

           1            2            3            4            5            6
0.0174011270 0.0768009809 0.0409503781 0.0215799628 0.0288029684 0.0010315088
           7            8            9           10           11           12
0.0031379129 0.0448481919 0.0188614178 0.1195191319 0.0958663105 0.0579850735
          13           14           15           16           17           18
0.0731523657 0.0002938362 0.0984796718 0.0919482890 0.0975367746 0.0546070811
          19           20           21           22           23           24
0.0182095530 0.0371812046 0.0107408856 0.0300692243 0.0053432866 0.0128138673
          25
0.0150803356
```

Capitolo 18

Regressione Probit

18.1 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 : $X_i = (1, x_{i1}, x_{i2}, \ldots, x_{ik-1}) \quad \forall\, i = 1, 2, \ldots, n$
- vettore numerico positivo dei pesi IWLS: $w = (w_1, w_2, \ldots, w_n)$
- 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 $n \times n$: $H = X\,(X^T\, W^{-1}\, X)^{-1}\, X^T\, W^{-1}$
- valori di leva: $h_i = H_{i,i} \quad \forall\, i = 1, 2, \ldots, n$
- distanza di *Cook*: $cd_i = \left(e_i^P\right)^2 \frac{h_i}{k\,(1-h_i)^2} \quad \forall\, i = 1, 2, \ldots, n$
- stime IWLS: $\hat{\beta} = \left(\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_k\right)^T$
- standard error delle stime IWLS: $s_{\hat{\beta}} = \sqrt{\text{diag}((X^T\, W^{-1}\, X)^{-1})}$
- z-values delle stime IWLS: $z_{\hat{\beta}} = \hat{\beta} \,/\, s_{\hat{\beta}}$
- correlazione delle stime IWLS: $r_{\hat{\beta}_i\,\hat{\beta}_j} = \frac{(X^T\, W^{-1}\, X)^{-1}_{i,\,j}}{s_{\hat{\beta}_i}\, s_{\hat{\beta}_j}} \quad \forall\, i, j = 1, 2, \ldots, k$
- residui di devianza: $e_i = \text{sign}\,(y_i - \hat{y}_i) \sqrt{2\left[y_i \log\left(\frac{y_i}{\hat{y}_i} + C_{i1}\right) + (n_i - y_i) \log\left(\frac{n_i - y_i}{n_i - \hat{y}_i} + C_{i2}\right)\right]}$

  $\forall\, i = 1, 2, \ldots, n$ dove $C_{i1} = 0.5\,(1 - \text{sign}(y_i)) \,/\, \hat{y}_i$ e $C_{i2} = 0.5\,(1 - \text{sign}(n_i - y_i)) \,/\, (n_i - \hat{y}_i)$
- residui standard: $rstandard_i = e_i \,/\, \sqrt{1 - h_i} \quad \forall\, i = 1, 2, \ldots, n$
- residui studentizzati: $rstudent_i = \text{sign}\,(y_i - \hat{y}_i) \sqrt{e_i^2 + h_i\,\left(e_i^P\right)^2 / (1 - h_i)} \quad \forall\, i = 1, 2, \ldots, n$
- residui di *Pearson*: $e_i^P = \frac{y_i - n_i\,\hat{\pi}_i}{\sqrt{n_i\,\hat{\pi}_i\,(1-\hat{\pi}_i)}} \quad \forall i = 1, 2, \ldots, n$
- residui di lavoro: $e_i^W = \frac{y_i - n_i\,\hat{\pi}_i}{n_i\,\hat{\pi}_i\,(1-\hat{\pi}_i)} \quad \forall i = 1, 2, \ldots, n$
- residui di riposta: $e_i^R = y_i \,/\, n_i - \hat{\pi}_i \quad \forall i = 1, 2, \ldots, n$
- log-verosimiglianza binomiale: $\hat{\ell} = \sum_{i=1}^{n} \left[\log\binom{n_i}{y_i} + y_i \log\left(\frac{\hat{y}_i}{n_i}\right) + (n_i - y_i) \log\left(1 - \frac{\hat{y}_i}{n_i}\right)\right]$
- valori adattati: $\hat{\pi}_i = \Phi\left(X_i\,\hat{\beta}\right) \quad \forall\, i = 1, 2, \ldots, n$

- numero di successi attesi: $\hat{y}_i = n_i \, \hat{\pi}_i \quad \forall \, i = 1, 2, \ldots, n$
- log-verosimiglianza binomiale modello saturo: $\hat{\ell}_{saturo} = \sum_{i=1}^n \left[\log \binom{n_i}{y_i} + y_i \, \log \left(\frac{y_i}{n_i} \right) + (n_i - y_i) \, \log \left(1 - \frac{y_i}{n_i} \right) \right]$
- devianza residua: $D = 2 \left(\hat{\ell}_{saturo} - \hat{\ell} \right) = \sum_{i=1}^n e_i^2$
- gradi di libertà della devianza residua: $n - k$
- log-verosimiglianza binomiale modello nullo: $\hat{\ell}_{nullo} = \sum_{i=1}^n \left[\log \binom{n_i}{y_i} + y_i \, \log (\hat{\pi}) + (n_i - y_i) \, \log (1 - \hat{\pi}) \right]$
- valori adattati modello nullo: $\hat{\pi} = \sum_{j=1}^n y_j \, / \, \sum_{j=1}^n n_j \quad \forall \, i = 1, 2, \ldots, n$
- numero di successi attesi modello nullo: $\hat{y}_i = n_i \, \hat{\pi} \quad \forall \, i = 1, 2, \ldots, n$
- devianza residua modello nullo: $D_{nullo} = 2 \left(\hat{\ell}_{saturo} - \hat{\ell}_{nullo} \right)$
- gradi di libertà della devianza residua modello nullo: $n - 1$
- stima IWLS intercetta modello nullo: $\hat{\beta}_{nullo} = \Phi^{-1} (\hat{\pi})$

18.2 Stima

glm()

- **Package:** `stats`
- **Input:**

  `formula` modello di regressione probit con $k - 1$ variabili esplicative ed n unità

  `family = binomial(link="probit")` famiglia e link del modello

  `x = TRUE` matrice del modello
- **Description:** analisi di regressione probit
- **Output:**

  `coefficients` stime IWLS

  `residuals` residui di lavoro

  `fitted.values` valori adattati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` 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`

  $$\hat{\beta}_j \quad \forall \, j = 1, 2, \ldots, k$$

  `residuals`

  $$e_i^W \quad \forall \, i = 1, 2, \ldots, n$$

  `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

$$X$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"),
+     x = TRUE)
> modello$coefficients

(Intercept)           x
 -11.818942    0.907823

> modello$residuals

           1            2            3            4            5            6
-0.269418259 -0.348625023 -0.389983219 -0.122461411 -0.200141756 -0.046955683
           7            8            9           10           11           12
-0.002815914  0.058111915 -0.133324114  0.140220542  0.121793589  0.102604272
          13           14           15           16           17           18
-0.118836507  0.054563070  0.218884846 -0.056123202 -0.104260350  0.228143827
          19           20           21           22           23           24
 0.136088873 -0.179601128 -0.148819712 -0.409392515 -0.420317445 -0.792660540
          25
 0.229368032

> modello$fitted.values
```

```
          1            2            3            4            5            6
0.0002722105 0.0053850922 0.0134084170 0.0234491271 0.0391816851 0.0626001924
          7            8            9           10           11           12
0.0957166773 0.1402058751 0.1969852207 0.2658269508 0.3451206813 0.4318871004
         13           14           15           16           17           18
0.5220837266 0.6111585001 0.6947274541 0.7692111098 0.8322781892 0.8830088002
         19           20           21           22           23           24
0.9217758718 0.9499195786 0.9693295476 0.9820468044 0.9899624601 0.9946430973
         25
0.9999826792
```

```
> modello$rank
```

```
[1] 2
```

```
> modello$linear.predictors
```

```
         1          2          3          4          5          6          7
-3.4578913 -2.5500682 -2.2141737 -1.9872179 -1.7602621 -1.5333064 -1.3063506
         8          9         10         11         12         13         14
-1.0793948 -0.8524391 -0.6254833 -0.3985275 -0.1715718  0.0553840  0.2823398
        15         16         17         18         19         20         21
 0.5092955  0.7362513  0.9632071  1.1901628  1.4171186  1.6440744  1.8710301
        22         23         24         25
 2.0979859  2.3249417  2.5518974  4.1405878
```

```
> modello$deviance
```

```
[1] 22.88743
```

```
> modello$aic
```

```
[1] 110.9392
```

```
> modello$null.deviance
```

```
[1] 3693.884
```

```
> modello$weights
```

```
         1          2          3          4          5          6          7
 1.4104551  8.9094789  8.3105953 16.0744621 17.1659357 22.7386165 35.0406005
         8          9         10         11         12         13         14
45.7076709 48.6499031 51.2857797 60.0774428 68.0228376 62.9551408 65.5510152
        15         16         17         18         19         20         21
60.7937719 60.9999288 44.1838731 36.2494196 35.5528528 22.8652682 19.7074642
        22         23         24         25
12.2829626  6.7637482  5.0575577  0.3453737
```

```
> modello$prior.weights
```

```
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
 376  200   93  120   90   88  105  111  100   93  100  108   99  106  105  117
  17   18   19   20   21   22   23   24   25
  98   97  120  102  122  111   94  114 1049
```

```
> modello$df.residual
```

```
[1] 23
```

```
> modello$df.null

[1] 24

> modello$y

         1          2          3          4          5          6          7
0.00000000 0.00000000 0.00000000 0.01666667 0.02222222 0.05681818 0.09523810
         8          9         10         11         12         13         14
0.15315315 0.16000000 0.31182796 0.39000000 0.47222222 0.47474747 0.63207547
        15         16         17         18         19         20         21
0.77142857 0.75213675 0.80612245 0.92783505 0.94166667 0.93137255 0.95901639
        22         23         24         25
0.96396396 0.97872340 0.98245614 1.00000000

> modello$x

   (Intercept)     x
1            1  9.21
2            1 10.21
3            1 10.58
4            1 10.83
5            1 11.08
6            1 11.33
7            1 11.58
8            1 11.83
9            1 12.08
10           1 12.33
11           1 12.58
12           1 12.83
13           1 13.08
14           1 13.33
15           1 13.58
16           1 13.83
17           1 14.08
18           1 14.33
19           1 14.58
20           1 14.83
21           1 15.08
22           1 15.33
23           1 15.58
24           1 15.83
25           1 17.58
attr(,"assign")
[1] 0 1
```

summary.glm()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k-1$ variabili esplicative ed n unità

  `correlation = TRUE` correlazione delle stime IWLS

- **Description:** 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 IWLS non scalata

`cov.scaled` matrice di covarianza delle stime IWLS scalata

`correlation` matrice di correlazione delle stime IWLS

- **Formula:**

`deviance`

$$D$$

`aic`

$$-2\,\hat{\ell} + 2\,k$$

`df.residual`

$$n - k$$

`null.deviance`

$$D_{nullo}$$

`df.null`

$$n - 1$$

`deviance.resid`

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

`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`

$$(X^T\,W^{-1}\,X)^{-1}$$

`correlation`

$$r_{\hat{\beta}_i\,\hat{\beta}_j} \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> res <- summary.glm(object = modello, correlation = TRUE)
> res$deviance

[1] 22.88743

> res$aic

[1] 110.9392

> res$df.residual

[1] 23
```

```
> res$null.deviance

[1] 3693.884

> res$df.null

[1] 24

> res$deviance.resid

          1           2           3           4           5           6
-0.45247119 -1.46964542 -1.58456196 -0.51743600 -0.90056726 -0.22725786
          7           8           9          10          11          12
-0.01668127  0.38801751 -0.95408459  0.98731872  0.93524092  0.84356724
         13          14          15          16          17          18
-0.94228925  0.44328398  1.75392860 -0.43468903 -0.67959504  1.46607128
         19          20          21          22          23          24
 0.84691681 -0.81514441 -0.62908579 -1.26364877 -0.95089420 -1.40845258
         25
 0.19062911

> res$coefficients

              Estimate Std. Error   z value      Pr(>|z|)
(Intercept) -11.818942 0.38701607 -30.53863 8.004674e-205
x             0.907823 0.02955339  30.71807 3.265395e-207

> res$cov.unscaled

            (Intercept)             x
(Intercept)  0.14978143 -0.0113907885
x           -0.01139079  0.0008734026

> res$cov.scaled

            (Intercept)             x
(Intercept)  0.14978143 -0.0113907885
x           -0.01139079  0.0008734026

> res$correlation

            (Intercept)          x
(Intercept)   1.0000000 -0.9959042
x            -0.9959042  1.0000000
```

glm.fit()

- **Package:** stats
- **Input:**

  x matrice del modello

  y proporzione di successi

  weights numero di prove

  family = binomial(link="probit") famiglia e link del modello
- **Description:** analisi di regressione probit
- **Output:**

`coefficients` stime IWLS
`residuals` residui di lavoro
`fitted.values` valori adattati
`rank` rango della matrice del modello
`linear.predictors` predittori lineari
`deviance` 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`
$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

`residuals`
$$e_i^W \quad \forall\, j = 1, 2, \ldots, n$$

`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`
$$y_i\,/\,n_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> X <- model.matrix(object = modello)
> res <- glm.fit(x = X, y/Total, weights = Total, family = binomial(link = "probit"))
> res$coefficients
```

```
(Intercept)           x
 -11.818942    0.907823

> res$residuals

 [1] -0.269418259 -0.348625023 -0.389983219 -0.122461411 -0.200141756
 [6] -0.046955683 -0.002815914  0.058111915 -0.133324114  0.140220542
[11]  0.121793589  0.102604272 -0.118836507  0.054563070  0.218884846
[16] -0.056123202 -0.104260350  0.228143827  0.136088873 -0.179601128
[21] -0.148819712 -0.409392515 -0.420317445 -0.792660540  0.229368032

> res$fitted.values

 [1] 0.0002722105 0.0053850922 0.0134084170 0.0234491271 0.0391816851
 [6] 0.0626001924 0.0957166773 0.1402058751 0.1969852207 0.2658269508
[11] 0.3451206813 0.4318871004 0.5220837266 0.6111585001 0.6947274541
[16] 0.7692111098 0.8322781892 0.8830088002 0.9217758718 0.9499195786
[21] 0.9693295476 0.9820468044 0.9899624601 0.9946430973 0.9999826792

> res$rank

[1] 2

> res$linear.predictors

 [1] -3.4578913 -2.5500682 -2.2141737 -1.9872179 -1.7602621 -1.5333064
 [7] -1.3063506 -1.0793948 -0.8524391 -0.6254833 -0.3985275 -0.1715718
[13]  0.0553840  0.2823398  0.5092955  0.7362513  0.9632071  1.1901628
[19]  1.4171186  1.6440744  1.8710301  2.0979859  2.3249417  2.5518974
[25]  4.1405878

> res$deviance

[1] 22.88743

> res$aic

[1] 110.9392

> res$null.deviance

[1] 3693.884

> res$weights

 [1]  1.4104551  8.9094789  8.3105953 16.0744621 17.1659357 22.7386165
 [7] 35.0406005 45.7076709 48.6499031 51.2857797 60.0774428 68.0228376
[13] 62.9551408 65.5510152 60.7937719 60.9999288 44.1838731 36.2494196
[19] 35.5528528 22.8652682 19.7074642 12.2829626  6.7637482  5.0575577
[25]  0.3453737

> res$prior.weights

 [1]  376  200   93  120   90   88  105  111  100   93  100  108   99  106  105
[16]  117   98   97  120  102  122  111   94  114 1049

> res$df.residual

[1] 23
```

```
> res$df.null

[1] 24

> res$y

 [1] 0.00000000 0.00000000 0.00000000 0.01666667 0.02222222 0.05681818
 [7] 0.09523810 0.15315315 0.16000000 0.31182796 0.39000000 0.47222222
[13] 0.47474747 0.63207547 0.77142857 0.75213675 0.80612245 0.92783505
[19] 0.94166667 0.93137255 0.95901639 0.96396396 0.97872340 0.98245614
[25] 1.00000000
```

vcov()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k - 1$ variabili esplicative ed n unità

- **Description:** matrice di covarianza delle stime IWLS
- **Formula:**

$$(X^T W^{-1} X)^{-1}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> vcov(object = modello)

             (Intercept)             x
(Intercept)  0.14978143 -0.0113907885
x           -0.01139079  0.0008734026
```

coef()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k - 1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> coef(object = modello)

(Intercept)          x
 -11.818942    0.907823
```

coefficients()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> coefficients(object = modello)

(Intercept)          x
 -11.818942    0.907823
```

predict.glm()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k-1$ variabili esplicative ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` standard error delle stime

- **Description:** previsione
- **Output:**

  `fit` valore previsto

  `se.fit` standard error delle stime

- **Formula:**

fit

$$x_0^T \, \hat{\beta}$$

se.fit

$$\sqrt{x_0^T \, (X^T \, W^{-1} \, X)^{-1} \, x_0}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> res <- predict.glm(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit

        1
-10.63877

> res$se.fit

[1] 0.3487713
```

predict()

- **Package:** stats
- **Input:**

  object modello di regressione probit con $k - 1$ variabili esplicative ed n unità

  newdata il valore di x_0

  se.fit = TRUE standard error delle stime

- **Description:** previsione
- **Output:**

  fit valore previsto

  se.fit standard error delle stime

- **Formula:**

fit

$$x_0^T \, \hat{\beta}$$

se.fit

$$\sqrt{x_0^T \, (X^T \, W^{-1} \, X)^{-1} \, x_0}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> res <- predict(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit
```

```
        1
-10.63877
```

```
> res$se.fit
```

```
[1] 0.3487713
```

fitted()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k - 1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> fitted(object = modello)
```

```
           1            2            3            4            5            6 
0.0002722105 0.0053850922 0.0134084170 0.0234491271 0.0391816851 0.0626001924 
           7            8            9           10           11           12 
0.0957166773 0.1402058751 0.1969852207 0.2658269508 0.3451206813 0.4318871004 
          13           14           15           16           17           18 
0.5220837266 0.6111585001 0.6947274541 0.7692111098 0.8322781892 0.8830088002 
          19           20           21           22           23           24 
0.9217758718 0.9499195786 0.9693295476 0.9820468044 0.9899624601 0.9946430973 
          25 
0.9999826792 
```

fitted.values()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k - 1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> fitted.values(object = modello)
```

```
           1            2            3            4            5            6
0.0002722105 0.0053850922 0.0134084170 0.0234491271 0.0391816851 0.0626001924
           7            8            9           10           11           12
0.0957166773 0.1402058751 0.1969852207 0.2658269508 0.3451206813 0.4318871004
          13           14           15           16           17           18
0.5220837266 0.6111585001 0.6947274541 0.7692111098 0.8322781892 0.8830088002
          19           20           21           22           23           24
0.9217758718 0.9499195786 0.9693295476 0.9820468044 0.9899624601 0.9946430973
          25
0.9999826792
```

cov2cor()

- **Package:** stats
- **Input:**

  V matrice di covarianza delle stime IWLS di dimensione $k \times k$

- **Description:** converte la matrice di covarianza nella matrice di correlazione
- **Formula:**

$$r_{\hat{\beta}_i \, \hat{\beta}_j} \quad \forall \, i, j \, = \, 1, \, 2, \, \ldots, \, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> V <- vcov(object = modello)
> cov2cor(V)
```

```
            (Intercept)          x
(Intercept)   1.0000000 -0.9959042
x            -0.9959042  1.0000000
```

18.3 Adattamento

logLik()

- **Package:** stats
- **Input:**

`object` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Description:** log-verosimiglianza binomiale
- **Formula:**

$$\hat{\ell}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> logLik(object = modello)

'log Lik.' -53.46962 (df=2)
```

AIC()

- **Package:** `stats`
- **Input:**

`object` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Description:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> AIC(object = modello)

[1] 110.9392
```

durbin.watson()

- **Package:** `car`
- **Input:**

`model` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Description:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

`dw` valore empirico della statistica *D–W*

- **Formula:**

  `dw`

$$\sum_{i=2}^{n} (e_i - e_{i-1})^2 \,/\, D$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> durbin.watson(model = modello)
```

```
 lag Autocorrelation D-W Statistic p-value
   1       0.3108564      1.367754    0.07
 Alternative hypothesis: rho != 0
```

```
> res <- durbin.watson(model = modello)
> res$dw
```

```
[1] 1.367754
```

extractAIC()

- **Package:** `stats`
- **Input:**

  `fit` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Description:** numero di parametri del modello ed indice *AIC* generalizzato
- **Formula:**

$$k \qquad -2\,\hat{\ell} + 2\,k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> extractAIC(fit = modello)
```

```
[1]   2.0000 110.9392
```

deviance()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Description:** devianza residua
- **Formula:**

$$D$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> deviance(object = modello)

[1] 22.88743
```

anova()

- **Package:** stats
- **Input:**

  `nullo` modello nullo di regressione probit con n unità

  `modello` modello di regressione probit con $k-1$ variabili esplicative con n unità

  `test = "Chisq"`

- **Description:** 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`

$$df = k-1$$

  `Deviance`

$$c = D_{nullo} - D$$

  `P(>|Chi|)`

$$P(\chi^2_{df} \geq c)$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> nullo <- glm(formula = cbind(y, Total - y) ~ 1, family = binomial(link = "probit"))
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> anova(nullo, modello, test = "Chisq")

Analysis of Deviance Table

Model 1: cbind(y, Total - y) ~ 1
Model 2: cbind(y, Total - y) ~ x
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1        24     3693.9
2        23       22.9  1   3671.0       0.0

> res <- anova(nullo, modello, test = "Chisq")
> res$"Resid. Df"

[1] 24 23

> res$"Resid. Dev"

[1] 3693.88357   22.88743

> res$Df

[1] NA  1

> res$Deviance

[1]       NA 3670.996

> res$"P(>|Chi|)"

[1] NA  0
```

drop1()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k-1$ variabili esplicative ed n unità

  `test = "Chisq"`
- **Description:** submodels
- **Output:**

  `Df` differenza tra gradi di libertà

  `Deviance` differenza tra devianze residue

  `AIC` indice AIC

  `LRT` valore empirico della statistica χ^2

  `Pr(Chi)` p-value

- **Formula:**

  Df

$$\underbrace{1,\,1,\,\dots,\,1}_{k-1 \text{ volte}}$$

  Deviance

$$D,\,D_{-x_j}\quad \forall\, j\,=\,1,\,2,\,\dots,\,k-1$$

  dove D_{-x_j} rappresenta la devianza residua del modello eliminata la variabile esplicativa x_j.

  AIC

$$-2\,\hat{\ell}+2\,k,\,-2\,\hat{\ell}_{-x_j}+2\,(k-1)\quad \forall\, j\,=\,1,\,2,\,\dots,\,k-1$$

  dove $\hat{\ell}_{-x_j}$ rappresenta la log-verosimiglianza binomiale del modello eliminata la variabile esplicativa x_j.

  LRT

$$c_j\,=\,D_{-x_j}-D\quad \forall\, j\,=\,1,\,2,\,\dots,\,k-1$$

  Pr(Chi)

$$P(\chi^2_1 \geq c_j)\quad \forall\, j\,=\,1,\,2,\,\dots,\,k-1$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> drop1(object = modello, test = "Chisq")

Single term deletions

Model:
cbind(y, Total - y) ~ x
       Df Deviance    AIC    LRT   Pr(Chi)
<none>        22.9  110.9
x       1   3693.9 3779.9 3671.0 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> res <- drop1(object = modello, test = "Chisq")
> res$Df

[1] NA  1

> res$Deviance

[1]   22.88743 3693.88357

> res$AIC

[1]  110.9392 3779.9354

> res$LRT

[1]       NA 3670.996

> res$"Pr(Chi)"

[1] NA  0
```

add1()

- **Package:** stats
- **Input:**

  `object` modello nullo di regressione probit

  `scope` modello di regressione probit con $k - 1$ variabili esplicative ed n unità

  `test = "Chisq"`
- **Description:** submodels
- **Output:**

  `Df` differenza tra gradi di libertà

  `Deviance` differenza tra devianze residue

  `AIC` indice AIC

  `LRT` valore empirico della statistica χ^2

  `Pr(Chi)` p-value
- **Formula:**

  `Df`

$$\underbrace{1, 1, \ldots, 1}_{k-1 \text{ volte}}$$

  `Deviance`

$$D_{nullo}, D_{x_j} \quad \forall\, j \,=\, 1, 2, \ldots, k-1$$

  dove D_{x_j} rappresenta la devianza residua del modello con la sola variabile esplicativa x_j.

  `AIC`

$$-2\,\hat{\ell}_{nullo} + 2, \; -2\,\hat{\ell}_{x_j} + 4 \quad \forall\, j \,=\, 1, 2, \ldots, k-1$$

  dove $\hat{\ell}_{x_j}$ rappresenta la log-verosimiglianza binomiale del modello con la sola variabile esplicativa x_j.

  `LRT`

$$c_j \,=\, D_{nullo} - D_{x_j} \quad \forall\, j \,=\, 1, 2, \ldots, k-1$$

  `Pr(Chi)`

$$P(\chi^2_1 \geq c_j) \quad \forall\, j \,=\, 1, 2, \ldots, k-1$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> nullo <- glm(formula = cbind(y, Total - y) ~ 1, family = binomial(link = "probit"))
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> add1(object = nullo, scope = modello, test = "Chisq")

Single term additions

Model:
cbind(y, Total - y) ~ 1
       Df Deviance    AIC    LRT   Pr(Chi)
<none>      3693.9 3779.9
x       1     22.9  110.9 3671.0 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> res <- add1(object = nullo, scope = modello, test = "Chisq")
> res$Df

[1] NA  1

> res$Deviance

[1] 3693.88357   22.88743

> res$AIC

[1] 3779.9354  110.9392

> res$LRT

[1]       NA 3670.996

> res$"Pr(Chi)"

[1] NA  0
```

18.4 Diagnostica

rstandard()

- **Package:** stats
- **Input:**

  model modello di regressione probit con $k-1$ variabili esplicative ed n unità
- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> rstandard(model = modello)

          1           2           3           4           5           6
-0.45702180 -1.52667261 -1.62930398 -0.54193441 -0.93825575 -0.23771437
          7           8           9          10          11          12
-0.01766532  0.41236338 -1.00506815  1.03243853  0.97758496  0.88234046
         13          14          15          16          17          18
-0.98089408  0.46342071  1.83843010 -0.46019719 -0.71464732  1.54273708
         19          20          21          22          23          24
 0.90128028 -0.85537455 -0.66151138 -1.31119403 -0.97372238 -1.43789404
         25
 0.19126471
```

rstandard.glm()

- **Package:** stats
- **Input:**

  model modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> rstandard.glm(model = modello)

          1           2           3           4           5           6
-0.45702180 -1.52667261 -1.62930398 -0.54193441 -0.93825575 -0.23771437
          7           8           9          10          11          12
-0.01766532  0.41236338 -1.00506815  1.03243853  0.97758496  0.88234046
         13          14          15          16          17          18
-0.98089408  0.46342071  1.83843010 -0.46019719 -0.71464732  1.54273708
         19          20          21          22          23          24
 0.90128028 -0.85537455 -0.66151138 -1.31119403 -0.97372238 -1.43789404
         25
 0.19126471
```

rstudent()

- **Package:** stats
- **Input:**

  model modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> rstudent(model = modello)
```

```
          1           2           3           4           5           6 
-0.45475250 -1.49850744 -1.60724034 -0.53954353 -0.93261903 -0.23741494 
          7           8           9          10          11          12 
-0.01766390  0.41295880 -1.00258075  1.03395739  0.97836584  0.88258097 
         13          14          15          16          17          18 
-0.98094312  0.46328566  1.83403420 -0.46061490 -0.71601113  1.53357601 
         19          20          21          22          23          24 
 0.89694597 -0.85968513 -0.66475785 -1.32462729 -0.98094946 -1.45532717 
         25 
 0.19094718 
```

rstudent.glm()

- **Package:** stats
- **Input:**

  `model` modello di regressione probit con $k - 1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> rstudent.glm(model = modello)
```

```
          1           2           3           4           5           6 
-0.45475250 -1.49850744 -1.60724034 -0.53954353 -0.93261903 -0.23741494 
          7           8           9          10          11          12 
-0.01766390  0.41295880 -1.00258075  1.03395739  0.97836584  0.88258097 
         13          14          15          16          17          18 
-0.98094312  0.46328566  1.83403420 -0.46061490 -0.71601113  1.53357601 
         19          20          21          22          23          24 
 0.89694597 -0.85968513 -0.66475785 -1.32462729 -0.98094946 -1.45532717 
         25 
 0.19094718 
```

residuals.default()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k - 1$ variabili esplicative ed n unità

- **Description:** residui di lavoro
- **Formula:**

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> residuals.default(object = modello)
```

```
           1            2            3            4            5            6
-0.269418259 -0.348625023 -0.389983219 -0.122461411 -0.200141756 -0.046955683
           7            8            9           10           11           12
-0.002815914  0.058111915 -0.133324114  0.140220542  0.121793589  0.102604272
          13           14           15           16           17           18
-0.118836507  0.054563070  0.218884846 -0.056123202 -0.104260350  0.228143827
          19           20           21           22           23           24
 0.136088873 -0.179601128 -0.148819712 -0.409392515 -0.420317445 -0.792660540
          25
 0.229368032
```

residuals()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo
- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> residuals(object = modello, type = "deviance")
```

```
           1           2           3           4           5           6
-0.45247119 -1.46964542 -1.58456196 -0.51743600 -0.90056726 -0.22725786
           7           8           9          10          11          12
-0.01668127  0.38801751 -0.95408459  0.98731872  0.93524092  0.84356724
          13          14          15          16          17          18
-0.94228925  0.44328398  1.75392860 -0.43468903 -0.67959504  1.46607128
          19          20          21          22          23          24
 0.84691681 -0.81514441 -0.62908579 -1.26364877 -0.95089420 -1.40845258
          25
 0.19062911
```

- **Example 2:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> residuals(object = modello, type = "pearson")

           1           2           3           4           5           6
-0.31996722 -1.04060064 -1.12424645 -0.49098375 -0.82922265 -0.22390818
           7           8           9          10          11          12
-0.01666883  0.39287973 -0.92992864  1.00417656  0.94401767  0.84623856
          13          14          15          16          17          18
-0.94289966  0.44176215  1.70665302 -0.43833594 -0.69302839  1.37359650
          19          20          21          22          23          24
 0.81144619 -0.85880990 -0.66065634 -1.43479933 -1.09312733 -1.78261348
          25
 0.13479572
```

- **Example 3:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> residuals(object = modello, type = "working")

            1            2            3            4            5            6
-0.269418259 -0.348625023 -0.389983219 -0.122461411 -0.200141756 -0.046955683
            7            8            9           10           11           12
-0.002815914  0.058111915 -0.133324114  0.140220542  0.121793589  0.102604272
           13           14           15           16           17           18
-0.118836507  0.054563070  0.218884846 -0.056123202 -0.104260350  0.228143827
           19           20           21           22           23           24
 0.136088873 -0.179601128 -0.148819712 -0.409392515 -0.420317445 -0.792660540
           25
 0.229368032
```

- **Example 4:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
```

```
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> residuals(object = modello, type = "response")

            1             2             3             4             5
-2.722105e-04 -5.385092e-03 -1.340842e-02 -6.782460e-03 -1.695946e-02
            6             7             8             9            10
-5.782011e-03 -4.785821e-04  1.294728e-02 -3.698522e-02  4.600101e-02
           11            12            13            14            15
 4.487932e-02  4.033512e-02 -4.733625e-02  2.091697e-02  7.670112e-02
           16            17            18            19            20
-1.707436e-02 -2.615574e-02  4.482625e-02  1.989079e-02 -1.854703e-02
           21            22            23            24            25
-1.031315e-02 -1.808284e-02 -1.123906e-02 -1.218696e-02  1.732085e-05
```

residuals.glm()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo

- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> residuals.glm(object = modello, type = "deviance")
```

```
          1           2           3           4           5           6
-0.45247119 -1.46964542 -1.58456196 -0.51743600 -0.90056726 -0.22725786
          7           8           9          10          11          12
-0.01668127  0.38801751 -0.95408459  0.98731872  0.93524092  0.84356724
         13          14          15          16          17          18
-0.94228925  0.44328398  1.75392860 -0.43468903 -0.67959504  1.46607128
         19          20          21          22          23          24
 0.84691681 -0.81514441 -0.62908579 -1.26364877 -0.95089420 -1.40845258
         25
 0.19062911
```

- **Example 2:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> residuals.glm(object = modello, type = "pearson")

          1           2           3           4           5           6
-0.31996722 -1.04060064 -1.12424645 -0.49098375 -0.82922265 -0.22390818
          7           8           9          10          11          12
-0.01666883  0.39287973 -0.92992864  1.00417656  0.94401767  0.84623856
         13          14          15          16          17          18
-0.94289966  0.44176215  1.70665302 -0.43833594 -0.69302839  1.37359650
         19          20          21          22          23          24
 0.81144619 -0.85880990 -0.66065634 -1.43479933 -1.09312733 -1.78261348
         25
 0.13479572
```

- **Example 3:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> residuals(object = modello, type = "working")

           1            2            3            4            5            6
-0.269418259 -0.348625023 -0.389983219 -0.122461411 -0.200141756 -0.046955683
           7            8            9           10           11           12
-0.002815914  0.058111915 -0.133324114  0.140220542  0.121793589  0.102604272
          13           14           15           16           17           18
-0.118836507  0.054563070  0.218884846 -0.056123202 -0.104260350  0.228143827
          19           20           21           22           23           24
 0.136088873 -0.179601128 -0.148819712 -0.409392515 -0.420317445 -0.792660540
          25
 0.229368032
```

- **Example 4:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
```

```
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+      88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+      108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+      1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> residuals.glm(object = modello, type = "response")

            1             2             3             4             5
-2.722105e-04 -5.385092e-03 -1.340842e-02 -6.782460e-03 -1.695946e-02
            6             7             8             9            10
-5.782011e-03 -4.785821e-04  1.294728e-02 -3.698522e-02  4.600101e-02
           11            12            13            14            15
 4.487932e-02  4.033512e-02 -4.733625e-02  2.091697e-02  7.670112e-02
           16            17            18            19            20
-1.707436e-02 -2.615574e-02  4.482625e-02  1.989079e-02 -1.854703e-02
           21            22            23            24            25
-1.031315e-02 -1.808284e-02 -1.123906e-02 -1.218696e-02  1.732085e-05
```

resid()

- **Package:** stats
- **Input:**

  `object` modello di regressione probit con $k - 1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo

- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+      12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+      14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+      88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+      108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+      1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> resid(object = modello, type = "deviance")
```

```
           1           2           3           4           5           6
-0.45247119 -1.46964542 -1.58456196 -0.51743600 -0.90056726 -0.22725786
           7           8           9          10          11          12
-0.01668127  0.38801751 -0.95408459  0.98731872  0.93524092  0.84356724
          13          14          15          16          17          18
-0.94228925  0.44328398  1.75392860 -0.43468903 -0.67959504  1.46607128
          19          20          21          22          23          24
 0.84691681 -0.81514441 -0.62908579 -1.26364877 -0.95089420 -1.40845258
          25
 0.19062911
```

- **Example 2:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> resid(object = modello, type = "pearson")

           1           2           3           4           5           6
-0.31996722 -1.04060064 -1.12424645 -0.49098375 -0.82922265 -0.22390818
           7           8           9          10          11          12
-0.01666883  0.39287973 -0.92992864  1.00417656  0.94401767  0.84623856
          13          14          15          16          17          18
-0.94289966  0.44176215  1.70665302 -0.43833594 -0.69302839  1.37359650
          19          20          21          22          23          24
 0.81144619 -0.85880990 -0.66065634 -1.43479933 -1.09312733 -1.78261348
          25
 0.13479572
```

- **Example 3:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> residuals(object = modello, type = "working")

            1            2            3            4            5            6
-0.269418259 -0.348625023 -0.389983219 -0.122461411 -0.200141756 -0.046955683
            7            8            9           10           11           12
-0.002815914  0.058111915 -0.133324114  0.140220542  0.121793589  0.102604272
           13           14           15           16           17           18
-0.118836507  0.054563070  0.218884846 -0.056123202 -0.104260350  0.228143827
           19           20           21           22           23           24
 0.136088873 -0.179601128 -0.148819712 -0.409392515 -0.420317445 -0.792660540
           25
 0.229368032
```

- **Example 4:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
```

```
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> resid(object = modello, type = "response")
```

```
            1             2             3             4             5
-2.722105e-04 -5.385092e-03 -1.340842e-02 -6.782460e-03 -1.695946e-02
            6             7             8             9            10
-5.782011e-03 -4.785821e-04  1.294728e-02 -3.698522e-02  4.600101e-02
           11            12            13            14            15
 4.487932e-02  4.033512e-02 -4.733625e-02  2.091697e-02  7.670112e-02
           16            17            18            19            20
-1.707436e-02 -2.615574e-02  4.482625e-02  1.989079e-02 -1.854703e-02
           21            22            23            24            25
-1.031315e-02 -1.808284e-02 -1.123906e-02 -1.218696e-02  1.732085e-05
```

weighted.residuals()

- **Package:** stats
- **Input:**

  obj modello di regressione probit con $k-1$ variabili esplicative ed n unità
- **Description:** residui pesati
- **Formula:**

$$e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> weighted.residuals(obj = modello)
```

```
          1           2           3           4           5           6
-0.45247119 -1.46964542 -1.58456196 -0.51743600 -0.90056726 -0.22725786
          7           8           9          10          11          12
-0.01668127  0.38801751 -0.95408459  0.98731872  0.93524092  0.84356724
         13          14          15          16          17          18
-0.94228925  0.44328398  1.75392860 -0.43468903 -0.67959504  1.46607128
         19          20          21          22          23          24
 0.84691681 -0.81514441 -0.62908579 -1.26364877 -0.95089420 -1.40845258
         25
 0.19062911
```

weights()

- **Package:** stats
- **Input:**

  object modello di regressione probit con $k - 1$ variabili esplicative ed n unità

- **Description:** pesi iniziali
- **Formula:**

$$n_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> weights(object = modello)

   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
 376  200   93  120   90   88  105  111  100   93  100  108   99  106  105  117
  17   18   19   20   21   22   23   24   25
  98   97  120  102  122  111   94  114 1049
```

df.residual()

- **Package:** stats
- **Input:**

  object modello di regressione probit con $k - 1$ variabili esplicative ed n unità

- **Description:** gradi di libertà della devianza residua
- **Formula:**

$$n - k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> df.residual(object = modello)

[1] 23
```

hatvalues()

- **Package:** stats
- **Input:**

  model modello di regressione probit con $k - 1$ variabili esplicative ed n unità

- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall \, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> hatvalues(model = modello)

          1           2           3           4           5           6
0.019815055 0.073312514 0.054167532 0.088367447 0.078723832 0.086040497
          7           8           9          10          11          12
0.108307417 0.114593994 0.098879759 0.085494466 0.084753718 0.085956150
         13          14          15          16          17          18
0.077164589 0.085016631 0.089815211 0.107785168 0.095690966 0.096919770
         19          20          21          22          23          24
0.116997841 0.091852356 0.095632164 0.071207217 0.046338837 0.040531561
         25
0.006635307
```

cooks.distance()

- **Package:** stats
- **Input:**

  model modello di regressione probit con $k - 1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall \, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> cooks.distance(model = modello)
```

```
             1              2              3              4              5              6
1.055748e-03 4.622210e-02 3.826517e-02 1.281613e-02 3.188885e-02 2.582016e-03
             7              8              9             10             11             12
1.892378e-05 1.128148e-02 5.265155e-02 5.154131e-02 4.508303e-02 3.683821e-02
            13             14             15             16             17             18
4.027824e-02 9.908879e-03 1.578888e-01 1.300781e-02 2.810019e-02 1.121110e-01
            19             20             21             22             23             24
4.940191e-02 4.107159e-02 2.551732e-02 8.496473e-02 3.044167e-02 6.995461e-02
            25
6.108938e-05
```

cookd()

- **Package:** car
- **Input:**

  model modello di regressione probit con $k-1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "probit"))
> cookd(model = modello)

             1              2              3              4              5              6
1.055748e-03 4.622210e-02 3.826517e-02 1.281613e-02 3.188885e-02 2.582016e-03
             7              8              9             10             11             12
1.892378e-05 1.128148e-02 5.265155e-02 5.154131e-02 4.508303e-02 3.683821e-02
            13             14             15             16             17             18
4.027824e-02 9.908879e-03 1.578888e-01 1.300781e-02 2.810019e-02 1.121110e-01
            19             20             21             22             23             24
4.940191e-02 4.107159e-02 2.551732e-02 8.496473e-02 3.044167e-02 6.995461e-02
            25
6.108938e-05
```

Capitolo 19

Regressione Log-log complementare

19.1 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 : $X_i \,=\, (1,\, x_{i1},\, x_{i2}, \ldots,\, x_{ik-1}) \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- vettore numerico positivo dei pesi IWLS: $w \,=\, (w_1,\, w_2,\, \ldots,\, w_n)$
- 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 $n \times n$: $H \,=\, X\, (X^T\, W^{-1}\, X)^{-1}\, X^T\, W^{-1}$
- valori di leva: $h_i \,=\, H_{i,i} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- distanza di *Cook*: $cd_i \,=\, \left(e_i^P\right)^2 \frac{h_i}{k\,(1-h_i)^2} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- stime IWLS: $\hat{\beta} \,=\, \left(\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_k\right)^T$
- standard error delle stime IWLS: $s_{\hat{\beta}} \,=\, \sqrt{\text{diag}((X^T\, W^{-1}\, X)^{-1})}$
- z-values delle stime IWLS: $z_{\hat{\beta}} \,=\, \hat{\beta} \,/\, s_{\hat{\beta}}$
- correlazione delle stime IWLS: $r_{\hat{\beta}_i\, \hat{\beta}_j} \,=\, \frac{(X^T\, W^{-1}\, X)^{-1}_{i,\,j}}{s_{\hat{\beta}_i}\, s_{\hat{\beta}_j}} \quad \forall\, i, j \,=\, 1,\, 2,\, \ldots,\, k$
- residui di devianza: $e_i \,=\, \text{sign}\,(y_i - \hat{y}_i) \sqrt{2 \left[y_i\, \log\left(\frac{y_i}{\hat{y}_i} + C_{i1}\right) + (n_i - y_i)\, \log\left(\frac{n_i - y_i}{n_i - \hat{y}_i} + C_{i2}\right)\right]}$

  $\forall\, i \,=\, 1,\, 2,\, \ldots,\, n$ dove $C_{i1} \,=\, 0.5\, (1 - \text{sign}(y_i))\, /\, \hat{y}_i$ e $C_{i2} \,=\, 0.5\, (1 - \text{sign}(n_i - y_i))\, /\, (n_i - \hat{y}_i)$
- residui standard: $rstandard_i \,=\, e_i \,/\, \sqrt{1 - h_i} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- residui studentizzati: $rstudent_i \,=\, \text{sign}\,(y_i - \hat{y}_i) \sqrt{e_i^2 + h_i\, \left(e_i^P\right)^2 / \,(1 - h_i)} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- residui di *Pearson*: $e_i^P \,=\, \frac{y_i - n_i\, \hat{\pi}_i}{\sqrt{n_i\, \hat{\pi}_i\, (1 - \hat{\pi}_i)}} \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$
- residui di lavoro: $e_i^W \,=\, \frac{y_i - n_i\, \hat{\pi}_i}{n_i\, \hat{\pi}_i\, (1 - \hat{\pi}_i)} \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$
- residui di riposta: $e_i^R \,=\, y_i \,/\, n_i - \hat{\pi}_i \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$
- log-verosimiglianza binomiale: $\hat{\ell} \,=\, \sum_{i=1}^n \left[\log \binom{n_i}{y_i} + y_i\, \log\left(\frac{\hat{y}_i}{n_i}\right) + (n_i - y_i)\, \log\left(1 - \frac{\hat{y}_i}{n_i}\right)\right]$
- valori adattati: $\hat{\pi}_i \,=\, 1 - \exp\left(-\exp\left(X_i\, \hat{\beta}\right)\right) \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$

- numero di successi attesi: $\hat{y}_i = n_i \, \hat{\pi}_i \quad \forall \, i = 1, 2, \ldots, n$
- log-verosimiglianza binomiale modello saturo: $\hat{\ell}_{saturo} = \sum_{i=1}^{n} \left[\log \binom{n_i}{y_i} + y_i \, \log \left(\frac{y_i}{n_i} \right) + (n_i - y_i) \, \log \left(1 - \frac{y_i}{n_i} \right) \right]$
- devianza residua: $D = 2 \left(\hat{\ell}_{saturo} - \hat{\ell} \right) = \sum_{i=1}^{n} e_i^2$
- gradi di libertà della devianza residua: $n - k$
- log-verosimiglianza binomiale modello nullo: $\hat{\ell}_{nullo} = \sum_{i=1}^{n} \left[\log \binom{n_i}{y_i} + y_i \, \log \left(\hat{\pi} \right) + (n_i - y_i) \, \log \left(1 - \hat{\pi} \right) \right]$
- valori adattati modello nullo: $\hat{\pi} = \sum_{j=1}^{n} y_j \, / \, \sum_{j=1}^{n} n_j \quad \forall \, i = 1, 2, \ldots, n$
- numero di successi attesi modello nullo: $\hat{y}_i = n_i \, \hat{\pi} \quad \forall \, i = 1, 2, \ldots, n$
- devianza residua modello nullo: $D_{nullo} = 2 \left(\hat{\ell}_{saturo} - \hat{\ell}_{nullo} \right)$
- gradi di libertà della devianza residua modello nullo: $n - 1$
- stima IWLS intercetta modello nullo: $\hat{\beta}_{nullo} = \log \left(- \log \left(1 - \hat{\pi} \right) \right)$

19.2 Stima

glm()

- **Package:** `stats`
- **Input:**

  `formula` modello di regressione log-log complementare con $k - 1$ variabili esplicative ed n unità

  `family = binomial(link="cloglog")` famiglia e link del modello

  `x = TRUE` matrice del modello

- **Description:** analisi di regressione log-log complementare
- **Output:**

  `coefficients` stime IWLS

  `residuals` residui di lavoro

  `fitted.values` valori adattati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` 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`

  $$\hat{\beta}_j \quad \forall \, j = 1, 2, \ldots, k$$

  `residuals`

  $$e_i^W \quad \forall \, i = 1, 2, \ldots, n$$

  `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

$$X$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"),
+     x = TRUE)
> modello$coefficients

(Intercept)           x
-12.9851164   0.9530076


> modello$residuals

          1           2           3           4           5           6
-1.00747570 -1.01954272 -1.02795778 -0.77915832 -0.77094233 -0.49169111
          7           8           9          10          11          12
-0.30341626 -0.08051823 -0.24628470  0.27292979  0.31833027  0.33451224
         13          14          15          16          17          18
 0.08077108  0.28820279  0.42232719  0.13526781  0.06070359  0.24992698
         19          20          21          22          23          24
 0.12113911 -0.19177587 -0.30930043 -0.93966307 -1.91670214 -7.49366104
         25
 1.00000000


> modello$fitted.values
```

```
         1          2          3          4          5          6          7
0.01476722 0.03784946 0.05341742 0.06729466 0.08461277 0.10612777 0.13270442
         8          9         10         11         12         13         14
0.16529635 0.20489911 0.25246255 0.30874773 0.37411551 0.44824630 0.52981661
        15         16         17         18         19         20         21
0.61620640 0.70337481 0.78609705 0.85873787 0.91656310 0.95722673 0.98168030
        22         23         24         25
0.99375413 0.99840579 0.99971820 1.00000000
```

```
> modello$rank
```

```
[1] 2
```

```
> modello$linear.predictors
```

```
          1           2           3           4           5           6
-4.20791595 -3.25490830 -2.90229547 -2.66404356 -2.42579164 -2.18753973
          7           8           9          10          11          12
-1.94928782 -1.71103591 -1.47278400 -1.23453209 -0.99628017 -0.75802826
         13          14          15          16          17          18
-0.51977635 -0.28152444 -0.04327253  0.19497939  0.43323130  0.67148321
         19          20          21          22          23          24
 0.90973512  1.14798703  1.38623894  1.62449086  1.86274277  2.10099468
         25
 3.76875806
```

```
> modello$deviance
```

```
[1] 118.8208
```

```
> modello$aic
```

```
[1] 206.8726
```

```
> modello$null.deviance
```

```
[1] 3693.884
```

```
> modello$weights
```

```
           1            2            3            4            5            6
5.551912e+00 7.568498e+00 4.966316e+00 8.071724e+00 7.609886e+00 9.329133e+00
           7            8            9           10           11           12
1.391005e+01 1.829764e+01 2.040002e+01 2.331378e+01 3.052613e+01 3.967311e+01
          13           14           15           16           17           18
4.309158e+01 5.356986e+01 5.997599e+01 7.287294e+01 6.342595e+01 6.111898e+01
          19           20           21           22           23           24
6.738325e+01 4.527553e+01 3.641982e+01 1.797138e+01 6.226026e+00 2.146377e+00
          25
2.329248e-13
```

```
> modello$prior.weights
```

```
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
 376  200   93  120   90   88  105  111  100   93  100  108   99  106  105  117
  17   18   19   20   21   22   23   24   25
  98   97  120  102  122  111   94  114 1049
```

```
> modello$df.residual
```

```
[1] 23

> modello$df.null

[1] 24

> modello$y

         1          2          3          4          5          6          7
0.00000000 0.00000000 0.00000000 0.01666667 0.02222222 0.05681818 0.09523810
         8          9         10         11         12         13         14
0.15315315 0.16000000 0.31182796 0.39000000 0.47222222 0.47474747 0.63207547
        15         16         17         18         19         20         21
0.77142857 0.75213675 0.80612245 0.92783505 0.94166667 0.93137255 0.95901639
        22         23         24         25
0.96396396 0.97872340 0.98245614 1.00000000

> modello$x

   (Intercept)     x
1            1  9.21
2            1 10.21
3            1 10.58
4            1 10.83
5            1 11.08
6            1 11.33
7            1 11.58
8            1 11.83
9            1 12.08
10           1 12.33
11           1 12.58
12           1 12.83
13           1 13.08
14           1 13.33
15           1 13.58
16           1 13.83
17           1 14.08
18           1 14.33
19           1 14.58
20           1 14.83
21           1 15.08
22           1 15.33
23           1 15.58
24           1 15.83
25           1 17.58
attr(,"assign")
[1] 0 1
```

summary.glm()

- **Package:** `stats`
- **Input:**

  `object` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

  `correlation = TRUE` correlazione delle stime IWLS

- **Description:** 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 IWLS non scalata

`cov.scaled` matrice di covarianza delle stime IWLS scalata

`correlation` matrice di correlazione delle stime IWLS

- **Formula:**

`deviance`

$$D$$

`aic`

$$-2\,\hat{\ell} + 2\,k$$

`df.residual`

$$n - k$$

`null.deviance`

$$D_{nullo}$$

`df.null`

$$n - 1$$

`deviance.resid`

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

`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`

$$(X^T\,W^{-1}\,X)^{-1}$$

`correlation`

$$r_{\hat{\beta}_i\,\hat{\beta}_j} \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> res <- summary.glm(object = modello, correlation = TRUE)
> res$deviance

[1] 118.8208

> res$aic

[1] 206.8726

> res$df.residual
```

```
[1] 23

> res$null.deviance

[1] 3693.884

> res$df.null

[1] 24

> res$deviance.resid

            1             2             3             4             5
-3.344811e+00 -3.928580e+00 -3.195443e+00 -2.625263e+00 -2.501326e+00
            6             7             8             9            10
-1.632697e+00 -1.183466e+00 -3.479272e-01 -1.146176e+00  1.287445e+00
           11            12            13            14            15
 1.722479e+00  2.078066e+00  5.293632e-01  2.125777e+00  3.393960e+00
           16            17            18            19            20
 1.175000e+00  4.892018e-01  2.127667e+00  1.046796e+00 -1.190182e+00
           21            22            23            24            25
-1.608195e+00 -2.739982e+00 -2.588698e+00 -3.552944e+00  6.825317e-07

> res$coefficients

               Estimate Std. Error   z value      Pr(>|z|)
(Intercept) -12.9851164 0.42631012 -30.45932 9.016015e-204
x             0.9530076 0.03133172  30.41671 3.303275e-203

> res$cov.unscaled

            (Intercept)             x
(Intercept)   0.1817403 -0.0133057991
x            -0.0133058  0.0009816765

> res$cov.scaled

            (Intercept)             x
(Intercept)   0.1817403 -0.0133057991
x            -0.0133058  0.0009816765

> res$correlation

            (Intercept)          x
(Intercept)   1.0000000 -0.9961646
x            -0.9961646  1.0000000
```

glm.fit()

- **Package:** `stats`
- **Input:**

  `x` matrice del modello

  `y` proporzione di successi

  `weights` numero di prove

  `family = binomial(link="cloglog")` famiglia e link del modello

- **Description:** analisi di regressione log-log complementare
- **Output:**

  `coefficients` stime IWLS

  `residuals` residui di lavoro

  `fitted.values` valori adattati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` 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`
  $$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

  `residuals`
  $$e_i^W \quad \forall\, i = 1, 2, \ldots, n$$

  `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

$$y_i \,/\, n_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> X <- model.matrix(object = modello)
> res <- glm.fit(x = X, y/Total, weights = Total, family = binomial(link = "cloglog"))
> res$coefficients

(Intercept)           x
-12.9851164   0.9530076

> res$residuals

 [1] -1.00747570 -1.01954272 -1.02795778 -0.77915832 -0.77094233 -0.49169111
 [7] -0.30341626 -0.08051823 -0.24628470  0.27292979  0.31833027  0.33451224
[13]  0.08077108  0.28820279  0.42232719  0.13526781  0.06070359  0.24992698
[19]  0.12113911 -0.19177587 -0.30930043 -0.93966307 -1.91670214 -7.49366104
[25]  1.00000000

> res$fitted.values

 [1] 0.01476722 0.03784946 0.05341742 0.06729466 0.08461277 0.10612777
 [7] 0.13270442 0.16529635 0.20489911 0.25246255 0.30874773 0.37411551
[13] 0.44824630 0.52981661 0.61620640 0.70337481 0.78609705 0.85873787
[19] 0.91656310 0.95722673 0.98168030 0.99375413 0.99840579 0.99971820
[25] 1.00000000

> res$rank

[1] 2

> res$linear.predictors

 [1] -4.20791595 -3.25490830 -2.90229547 -2.66404356 -2.42579164 -2.18753973
 [7] -1.94928782 -1.71103591 -1.47278400 -1.23453209 -0.99628017 -0.75802826
[13] -0.51977635 -0.28152444 -0.04327253  0.19497939  0.43323130  0.67148321
[19]  0.90973512  1.14798703  1.38623894  1.62449086  1.86274277  2.10099468
[25]  3.76875806

> res$deviance

[1] 118.8208

> res$aic

[1] 206.8726

> res$null.deviance

[1] 3693.884
```

```
> res$weights
```

```
 [1] 5.551912e+00 7.568498e+00 4.966316e+00 8.071724e+00 7.609886e+00
 [6] 9.329133e+00 1.391005e+01 1.829764e+01 2.040002e+01 2.331378e+01
[11] 3.052613e+01 3.967311e+01 4.309158e+01 5.356986e+01 5.997599e+01
[16] 7.287294e+01 6.342595e+01 6.111898e+01 6.738325e+01 4.527553e+01
[21] 3.641982e+01 1.797138e+01 6.226026e+00 2.146377e+00 2.329248e-13
```

```
> res$prior.weights
```

```
 [1]  376  200   93  120   90   88  105  111  100   93  100  108   99  106  105
[16]  117   98   97  120  102  122  111   94  114 1049
```

```
> res$df.residual
```

```
[1] 23
```

```
> res$df.null
```

```
[1] 24
```

```
> res$y
```

```
 [1] 0.00000000 0.00000000 0.00000000 0.01666667 0.02222222 0.05681818
 [7] 0.09523810 0.15315315 0.16000000 0.31182796 0.39000000 0.47222222
[13] 0.47474747 0.63207547 0.77142857 0.75213675 0.80612245 0.92783505
[19] 0.94166667 0.93137255 0.95901639 0.96396396 0.97872340 0.98245614
[25] 1.00000000
```

vcov()

- **Package:** stats
- **Input:**

  `object` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Description:** matrice di covarianza delle stime IWLS
- **Formula:**

$$(X^T W^{-1} X)^{-1}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> vcov(object = modello)
```

```
            (Intercept)             x
(Intercept)   0.1817403 -0.0133057991
x            -0.0133058  0.0009816765
```

coef()

- **Package:** stats
- **Input:**

  `object` modello di regressione log-log complementare con $k - 1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> coef(object = modello)

(Intercept)           x
-12.9851164   0.9530076
```

coefficients()

- **Package:** stats
- **Input:**

  `object` modello di regressione log-log complementare con $k - 1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> coefficients(object = modello)

(Intercept)           x
-12.9851164   0.9530076
```

predict.glm()

- **Package:** stats
- **Input:**

  object modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

  newdata il valore di x_0

  se.fit = TRUE standard error delle stime

- **Description:** previsione
- **Output:**

  fit valore previsto

  se.fit standard error delle stime

- **Formula:**

  fit

$$x_0^T \, \hat{\beta}$$

  se.fit

$$\sqrt{x_0^T \, (X^T \, W^{-1} \, X)^{-1} \, x_0}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> predict.glm(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)

$fit
        1
-11.74621

$se.fit
[1] 0.3857516

$residual.scale
[1] 1


> res <- predict.glm(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit


        1
-11.74621


> res$se.fit


[1] 0.3857516
```

predict()

- **Package:** stats
- **Input:**

  `object` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` standard error delle stime

- **Description:** previsione
- **Output:**

  `fit` valore previsto

  `se.fit` standard error delle stime

- **Formula:**

  `fit`

$$x_0^T \, \hat{\beta}$$

  `se.fit`

$$\sqrt{x_0^T \, (X^T \, W^{-1} \, X)^{-1} \, x_0}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> predict.glm(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)

$fit
        1
-11.74621

$se.fit
[1] 0.3857516

$residual.scale
[1] 1


> res <- predict.glm(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit


        1
-11.74621


> res$se.fit


[1] 0.3857516
```

fitted()

- **Package:** stats
- **Input:**

  object modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> fitted(object = modello)

         1          2          3          4          5          6          7
0.01476722 0.03784946 0.05341742 0.06729466 0.08461277 0.10612777 0.13270442
         8          9         10         11         12         13         14
0.16529635 0.20489911 0.25246255 0.30874773 0.37411551 0.44824630 0.52981661
        15         16         17         18         19         20         21
0.61620640 0.70337481 0.78609705 0.85873787 0.91656310 0.95722673 0.98168030
        22         23         24         25
0.99375413 0.99840579 0.99971820 1.00000000
```

fitted.values()

- **Package:** stats
- **Input:**

  object modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> fitted.values(object = modello)
```

```
         1          2          3          4          5          6          7
0.01476722 0.03784946 0.05341742 0.06729466 0.08461277 0.10612777 0.13270442
         8          9         10         11         12         13         14
0.16529635 0.20489911 0.25246255 0.30874773 0.37411551 0.44824630 0.52981661
        15         16         17         18         19         20         21
0.61620640 0.70337481 0.78609705 0.85873787 0.91656310 0.95722673 0.98168030
        22         23         24         25
0.99375413 0.99840579 0.99971820 1.00000000
```

cov2cor()

- **Package:** stats
- **Input:**

  V matrice di covarianza delle stime IWLS di dimensione $k \times k$

- **Description:** converte la matrice di covarianza nella matrice di correlazione
- **Formula:**

$$r_{\hat{\beta}_i \, \hat{\beta}_j} \quad \forall \, i, j \, = \, 1, \, 2, \, \ldots, \, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> V <- vcov(object = modello)
> cov2cor(V)

            (Intercept)          x
(Intercept)   1.0000000 -0.9961646
x            -0.9961646  1.0000000
```

19.3 Adattamento

logLik()

- **Package:** stats
- **Input:**

  object modello di regressione log-log complementare con $k - 1$ variabili esplicative ed n unità

- **Description:** log-verosimiglianza binomiale
- **Formula:**

$$\hat{\ell}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> logLik(object = modello)

'log Lik.' -101.4363 (df=2)
```

AIC()

- **Package:** stats
- **Input:**

  `object` modello di regressione log-log complementare con $k - 1$ variabili esplicative ed n unità

- **Description:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> AIC(object = modello)

[1] 206.8726
```

durbin.watson()

- **Package:** car
- **Input:**

  `model` modello di regressione cloglog con $k - 1$ variabili esplicative ed n unità

- **Description:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

  `dw` valore empirico della statistica *D–W*

- **Formula:**

  `dw`

$$\sum_{i=2}^{n} (e_i - e_{i-1})^2 \,/\, D$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> durbin.watson(model = modello)

 lag Autocorrelation D-W Statistic p-value
   1       0.7610921     0.3836592       0
 Alternative hypothesis: rho != 0

> res <- durbin.watson(model = modello)
> res$dw

[1] 0.3836592
```

extractAIC()

- **Package:** stats
- **Input:**

  `fit` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Description:** numero di parametri del modello ed indice *AIC* generalizzato
- **Formula:**

$$k \qquad -2\,\hat{\ell} + 2\,k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> extractAIC(fit = modello)

[1]   2.0000 206.8726
```

deviance()

- **Package:** stats
- **Input:**

  `object` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Description:** devianza residua
- **Formula:**

$$D$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> deviance(object = modello)

[1] 118.8208
```

anova()

- **Package:** stats
- **Input:**

  `nullo` modello nullo di regressione log-log complementare con n unità

  `modello` modello di regressione log-log complementare con $k-1$ variabili esplicative con n unità

  `test = "Chisq"`
- **Description:** 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`

  $$df = k-1$$

  `Deviance`

  $$c = D_{nullo} - D$$

  `P(>|Chi|)`

  $$P(\chi^2_{df} \geq c)$$
- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> nullo <- glm(formula = cbind(y, Total - y) ~ 1, family = binomial(link = "cloglog"))
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> anova(nullo, modello, test = "Chisq")
```

```
Analysis of Deviance Table

Model 1: cbind(y, Total - y) ~ 1
Model 2: cbind(y, Total - y) ~ x
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1        24     3693.9
2        23      118.8  1   3575.1       0.0

> res <- anova(nullo, modello, test = "Chisq")
> res$"Resid. Df"

[1] 24 23

> res$"Resid. Dev"

[1] 3693.8836  118.8208

> res$Df

[1] NA  1

> res$Deviance

[1]       NA 3575.063

> res$"P(>|Chi|)"

[1] NA  0
```

drop1()

- **Package:** stats
- **Input:**

  `object` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

  `test = "Chisq"`
- **Description:** submodels
- **Output:**

  `Df` differenza tra gradi di libertà

  `Deviance` differenza tra devianze residue

  `AIC` indice AIC

  `LRT` valore empirico della statistica χ^2

  `Pr(Chi)` p-value
- **Formula:**

  `Df`

$$\underbrace{1,\,1,\,\ldots,\,1}_{k-1 \text{ volte}}$$

  `Deviance`

$$D,\, D_{-x_j} \quad \forall\, j \,=\, 1,\,2,\,\ldots,\,k-1$$

  dove D_{-x_j} rappresenta la devianza residua del modello eliminata la variabile esplicativa x_j.

AIC

$$-2\,\hat{\ell} + 2\,k,\; -2\,\hat{\ell}_{-x_j} + 2\,(k-1) \quad \forall\, j \,=\, 1,\,2,\,\ldots,\,k-1$$

dove $\hat{\ell}_{-x_j}$ rappresenta la log-verosimiglianza binomiale del modello eliminata la variabile esplicativa x_j.

LRT

$$c_j \,=\, D_{-x_j} - D \quad \forall\, j \,=\, 1,\,2,\,\ldots,\,k-1$$

Pr(Chi)

$$P(\chi^2_1 \geq c_j) \quad \forall\, j \,=\, 1,\,2,\,\ldots,\,k-1$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> drop1(object = modello, test = "Chisq")


Single term deletions

Model:
cbind(y, Total - y) ~ x
       Df Deviance    AIC    LRT   Pr(Chi)
<none>       118.8  206.9
x       1   3693.9 3779.9 3575.1 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1


> res <- drop1(object = modello, test = "Chisq")
> res$Df


[1] NA  1


> res$Deviance


[1]  118.8208 3693.8836


> res$AIC


[1]  206.8726 3779.9354


> res$LRT


[1]       NA 3575.063


> res$"Pr(Chi)"


[1] NA  0
```

add1()

- **Package:** stats
- **Input:**

  `object` modello nullo di regressione log-log complementare

  `scope` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

  `test = "Chisq"`
- **Description:** submodels
- **Output:**

  `Df` differenza tra gradi di libertà

  `Deviance` differenza tra devianze residue

  `AIC` indice AIC

  `LRT` valore empirico della statistica χ^2

  `Pr(Chi)` p-value
- **Formula:**

  `Df`

$$\underbrace{1,\,1,\,\ldots,\,1}_{k-1 \text{ volte}}$$

  `Deviance`

$$D_{nullo},\, D_{x_j} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  dove D_{x_j} rappresenta la devianza residua del modello con la sola variabile esplicativa x_j.

  `AIC`

$$-2\,\hat{\ell}_{nullo} + 2,\; -2\,\hat{\ell}_{x_j} + 4 \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  dove $\hat{\ell}_{x_j}$ rappresenta la log-verosimiglianza binomiale del modello con la sola variabile esplicativa x_j.

  `LRT`

$$c_j \,=\, D_{nullo} - D_{x_j} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  `Pr(Chi)`

$$P(\chi^2_1 \geq c_j) \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$
- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> nullo <- glm(formula = cbind(y, Total - y) ~ 1, family = binomial(link = "cloglog"))
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> add1(object = nullo, scope = modello, test = "Chisq")

Single term additions

Model:
cbind(y, Total - y) ~ 1
       Df Deviance    AIC    LRT   Pr(Chi)
<none>      3693.9 3779.9
x       1    118.8  206.9 3575.1 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> res <- add1(object = nullo, scope = modello, test = "Chisq")
> res$Df

[1] NA  1

> res$Deviance

[1] 3693.8836  118.8208

> res$AIC

[1] 3779.9354  206.8726

> res$LRT

[1]       NA 3575.063

> res$"Pr(Chi)"

[1] NA  0
```

19.4 Diagnostica

rstandard()

- **Package:** stats
- **Input:**

  model modello di regressione log-log complementare con $k - 1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> rstandard(model = modello)

            1             2             3             4             5
-3.546647e+00 -4.126490e+00 -3.278516e+00 -2.722320e+00 -2.574884e+00
            6             7             8             9            10
-1.682464e+00 -1.228898e+00 -3.625140e-01 -1.189748e+00  1.332682e+00
           11            12            13            14            15
 1.787005e+00  2.161401e+00  5.487673e-01  2.212887e+00  3.545180e+00
           16            17            18            19            20
 1.243292e+00  5.172376e-01  2.269593e+00  1.144446e+00 -1.279947e+00
           21            22            23            24            25
-1.728057e+00 -2.857626e+00 -2.633515e+00 -3.577897e+00  6.825317e-07
```

rstandard.glm()

- **Package:** stats
- **Input:**

  `model` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+      12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+      14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+      88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+      108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+      1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> rstandard.glm(model = modello)

            1             2             3             4             5
-3.546647e+00 -4.126490e+00 -3.278516e+00 -2.722320e+00 -2.574884e+00
            6             7             8             9            10
-1.682464e+00 -1.228898e+00 -3.625140e-01 -1.189748e+00  1.332682e+00
           11            12            13            14            15
 1.787005e+00  2.161401e+00  5.487673e-01  2.212887e+00  3.545180e+00
           16            17            18            19            20
 1.243292e+00  5.172376e-01  2.269593e+00  1.144446e+00 -1.279947e+00
           21            22            23            24            25
-1.728057e+00 -2.857626e+00 -2.633515e+00 -3.577897e+00  6.825317e-07
```

rstudent()

- **Package:** stats
- **Input:**

  `model` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+      12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+      14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+      88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+      108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+      1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> rstudent(model = modello)
```

```
              1               2               3               4               5 
-3.447960e+00 -4.030684e+00 -3.238407e+00 -2.694633e+00 -2.554716e+00 
              6               7               8               9              10 
-1.674902e+00 -1.225072e+00 -3.622277e-01 -1.187261e+00  1.334804e+00 
             11              12              13              14              15 
 1.789702e+00  2.163690e+00  5.488287e-01  2.211575e+00  3.534607e+00 
             16              17              18              19              20 
 1.241017e+00  5.165991e-01  2.247950e+00  1.135287e+00 -1.295065e+00 
             21              22              23              24              25 
-1.767784e+00 -2.983221e+00 -2.738686e+00 -3.784579e+00  6.825317e-07 
```

rstudent.glm()

- **Package:** stats
- **Input:**

  `model` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità
- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> rstudent.glm(model = modello)

              1               2               3               4               5 
-3.447960e+00 -4.030684e+00 -3.238407e+00 -2.694633e+00 -2.554716e+00 
              6               7               8               9              10 
-1.674902e+00 -1.225072e+00 -3.622277e-01 -1.187261e+00  1.334804e+00 
             11              12              13              14              15 
 1.789702e+00  2.163690e+00  5.488287e-01  2.211575e+00  3.534607e+00 
             16              17              18              19              20 
 1.241017e+00  5.165991e-01  2.247950e+00  1.135287e+00 -1.295065e+00 
             21              22              23              24              25 
-1.767784e+00 -2.983221e+00 -2.738686e+00 -3.784579e+00  6.825317e-07 
```

residuals.default()

- **Package:** stats
- **Input:**

  `object` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità
- **Description:** residui di lavoro
- **Formula:**

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> residuals.default(object = modello)

          1           2           3           4           5           6
-1.00747570 -1.01954272 -1.02795778 -0.77915832 -0.77094233 -0.49169111
          7           8           9          10          11          12
-0.30341626 -0.08051823 -0.24628470  0.27292979  0.31833027  0.33451224
         13          14          15          16          17          18
 0.08077108  0.28820279  0.42232719  0.13526781  0.06070359  0.24992698
         19          20          21          22          23          24
 0.12113911 -0.19177587 -0.30930043 -0.93966307 -1.91670214 -7.49366104
         25
 1.00000000
```

residuals()

- **Package:** `stats`
- **Input:**

  `object` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo
- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> residuals(object = modello, type = "deviance")
```

```
            1             2             3             4             5
-3.344811e+00 -3.928580e+00 -3.195443e+00 -2.625263e+00 -2.501326e+00
            6             7             8             9            10
-1.632697e+00 -1.183466e+00 -3.479272e-01 -1.146176e+00  1.287445e+00
           11            12            13            14            15
 1.722479e+00  2.078066e+00  5.293632e-01  2.125777e+00  3.393960e+00
           16            17            18            19            20
 1.175000e+00  4.892018e-01  2.127667e+00  1.046796e+00 -1.190182e+00
           21            22            23            24            25
-1.608195e+00 -2.739982e+00 -2.588698e+00 -3.552944e+00  6.825317e-07
```

- **Example 2:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> residuals(object = modello, type = "pearson")

            1             2             3             4             5
-2.373963e+00 -2.804939e+00 -2.290887e+00 -2.213700e+00 -2.126766e+00
            6             7             8             9            10
-1.501829e+00 -1.131643e+00 -3.444267e-01 -1.112389e+00  1.317832e+00
           11            12            13            14            15
 1.758796e+00  2.106981e+00  5.302147e-01  2.109393e+00  3.270668e+00
           16            17            18            19            20
 1.154719e+00  4.834456e-01  1.953903e+00  9.944108e-01 -1.290438e+00
           21            22            23            24            25
-1.866683e+00 -3.983806e+00 -4.783173e+00 -1.098075e+01  4.826228e-07
```

- **Example 3:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> residuals(object = modello, type = "working")

          1           2           3           4           5           6
-1.00747570 -1.01954272 -1.02795778 -0.77915832 -0.77094233 -0.49169111
          7           8           9          10          11          12
-0.30341626 -0.08051823 -0.24628470  0.27292979  0.31833027  0.33451224
         13          14          15          16          17          18
 0.08077108  0.28820279  0.42232719  0.13526781  0.06070359  0.24992698
         19          20          21          22          23          24
 0.12113911 -0.19177587 -0.30930043 -0.93966307 -1.91670214 -7.49366104
         25
 1.00000000
```

- **Example 4:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
```

```
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> residuals(object = modello, type = "response")

            1             2             3             4             5
-1.476722e-02 -3.784946e-02 -5.341742e-02 -5.062800e-02 -6.239055e-02
            6             7             8             9            10
-4.930959e-02 -3.746632e-02 -1.214320e-02 -4.489911e-02  5.936540e-02
           11            12            13            14            15
 8.125227e-02  9.810671e-02  2.650118e-02  1.022589e-01  1.552222e-01
           16            17            18            19            20
 4.876194e-02  2.002539e-02  6.909718e-02  2.510357e-02 -2.585418e-02
           21            22            23            24            25
-2.266391e-02 -2.979016e-02 -1.968239e-02 -1.726206e-02  2.220446e-16
```

residuals.glm()

- **Package:** stats
- **Input:**

  `object` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo

- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> residuals.glm(object = modello, type = "deviance")
```

```
            1             2             3             4             5
-3.344811e+00 -3.928580e+00 -3.195443e+00 -2.625263e+00 -2.501326e+00
            6             7             8             9            10
-1.632697e+00 -1.183466e+00 -3.479272e-01 -1.146176e+00  1.287445e+00
           11            12            13            14            15
 1.722479e+00  2.078066e+00  5.293632e-01  2.125777e+00  3.393960e+00
           16            17            18            19            20
 1.175000e+00  4.892018e-01  2.127667e+00  1.046796e+00 -1.190182e+00
           21            22            23            24            25
-1.608195e+00 -2.739982e+00 -2.588698e+00 -3.552944e+00  6.825317e-07
```

- **Example 2:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> residuals.glm(object = modello, type = "pearson")

            1             2             3             4             5
-2.373963e+00 -2.804939e+00 -2.290887e+00 -2.213700e+00 -2.126766e+00
            6             7             8             9            10
-1.501829e+00 -1.131643e+00 -3.444267e-01 -1.112389e+00  1.317832e+00
           11            12            13            14            15
 1.758796e+00  2.106981e+00  5.302147e-01  2.109393e+00  3.270668e+00
           16            17            18            19            20
 1.154719e+00  4.834456e-01  1.953903e+00  9.944108e-01 -1.290438e+00
           21            22            23            24            25
-1.866683e+00 -3.983806e+00 -4.783173e+00 -1.098075e+01  4.826228e-07
```

- **Example 3:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> residuals.glm(object = modello, type = "working")

          1           2           3           4           5           6
-1.00747570 -1.01954272 -1.02795778 -0.77915832 -0.77094233 -0.49169111
          7           8           9          10          11          12
-0.30341626 -0.08051823 -0.24628470  0.27292979  0.31833027  0.33451224
         13          14          15          16          17          18
 0.08077108  0.28820279  0.42232719  0.13526781  0.06070359  0.24992698
         19          20          21          22          23          24
 0.12113911 -0.19177587 -0.30930043 -0.93966307 -1.91670214 -7.49366104
         25
 1.00000000
```

- **Example 4:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
```

```
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> residuals.glm(object = modello, type = "response")

            1             2             3             4             5
-1.476722e-02 -3.784946e-02 -5.341742e-02 -5.062800e-02 -6.239055e-02
            6             7             8             9            10
-4.930959e-02 -3.746632e-02 -1.214320e-02 -4.489911e-02  5.936540e-02
           11            12            13            14            15
 8.125227e-02  9.810671e-02  2.650118e-02  1.022589e-01  1.552222e-01
           16            17            18            19            20
 4.876194e-02  2.002539e-02  6.909718e-02  2.510357e-02 -2.585418e-02
           21            22            23            24            25
-2.266391e-02 -2.979016e-02 -1.968239e-02 -1.726206e-02  2.220446e-16
```

resid()

- **Package:** stats
- **Input:**

  `object` modello di regressione log-log complementare con $k - 1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo
- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> resid(object = modello, type = "deviance")
```

```
            1             2             3             4             5
-3.344811e+00 -3.928580e+00 -3.195443e+00 -2.625263e+00 -2.501326e+00
            6             7             8             9            10
-1.632697e+00 -1.183466e+00 -3.479272e-01 -1.146176e+00  1.287445e+00
           11            12            13            14            15
 1.722479e+00  2.078066e+00  5.293632e-01  2.125777e+00  3.393960e+00
           16            17            18            19            20
 1.175000e+00  4.892018e-01  2.127667e+00  1.046796e+00 -1.190182e+00
           21            22            23            24            25
-1.608195e+00 -2.739982e+00 -2.588698e+00 -3.552944e+00  6.825317e-07
```

- **Example 2:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> resid(object = modello, type = "pearson")

            1             2             3             4             5
-2.373963e+00 -2.804939e+00 -2.290887e+00 -2.213700e+00 -2.126766e+00
            6             7             8             9            10
-1.501829e+00 -1.131643e+00 -3.444267e-01 -1.112389e+00  1.317832e+00
           11            12            13            14            15
 1.758796e+00  2.106981e+00  5.302147e-01  2.109393e+00  3.270668e+00
           16            17            18            19            20
 1.154719e+00  4.834456e-01  1.953903e+00  9.944108e-01 -1.290438e+00
           21            22            23            24            25
-1.866683e+00 -3.983806e+00 -4.783173e+00 -1.098075e+01  4.826228e-07
```

- **Example 3:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> resid(object = modello, type = "working")

          1           2           3           4           5           6
-1.00747570 -1.01954272 -1.02795778 -0.77915832 -0.77094233 -0.49169111
          7           8           9          10          11          12
-0.30341626 -0.08051823 -0.24628470  0.27292979  0.31833027  0.33451224
         13          14          15          16          17          18
 0.08077108  0.28820279  0.42232719  0.13526781  0.06070359  0.24992698
         19          20          21          22          23          24
 0.12113911 -0.19177587 -0.30930043 -0.93966307 -1.91670214 -7.49366104
         25
 1.00000000
```

- **Example 4:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
```

```
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> resid(object = modello, type = "response")
```

```
            1             2             3             4             5
-1.476722e-02 -3.784946e-02 -5.341742e-02 -5.062800e-02 -6.239055e-02
            6             7             8             9            10
-4.930959e-02 -3.746632e-02 -1.214320e-02 -4.489911e-02  5.936540e-02
           11            12            13            14            15
 8.125227e-02  9.810671e-02  2.650118e-02  1.022589e-01  1.552222e-01
           16            17            18            19            20
 4.876194e-02  2.002539e-02  6.909718e-02  2.510357e-02 -2.585418e-02
           21            22            23            24            25
-2.266391e-02 -2.979016e-02 -1.968239e-02 -1.726206e-02  2.220446e-16
```

weighted.residuals()

- **Package:** stats
- **Input:**

  obj modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Description:** residui pesati
- **Formula:**

$$e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> weighted.residuals(obj = modello)
```

```
            1             2             3             4             5
-3.344811e+00 -3.928580e+00 -3.195443e+00 -2.625263e+00 -2.501326e+00
            6             7             8             9            10
-1.632697e+00 -1.183466e+00 -3.479272e-01 -1.146176e+00  1.287445e+00
           11            12            13            14            15
 1.722479e+00  2.078066e+00  5.293632e-01  2.125777e+00  3.393960e+00
           16            17            18            19            20
 1.175000e+00  4.892018e-01  2.127667e+00  1.046796e+00 -1.190182e+00
           21            22            23            24            25
-1.608195e+00 -2.739982e+00 -2.588698e+00 -3.552944e+00  6.825317e-07
```

weights()

- **Package:** stats
- **Input:**

  object modello di regressione log-log complementare con $k - 1$ variabili esplicative ed n unità

- **Description:** pesi iniziali
- **Formula:**

$$n_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> weights(object = modello)

   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
 376  200   93  120   90   88  105  111  100   93  100  108   99  106  105  117
  17   18   19   20   21   22   23   24   25
  98   97  120  102  122  111   94  114 1049
```

df.residual()

- **Package:** stats
- **Input:**

  object modello di regressione log-log complementare con $k - 1$ variabili esplicative ed n unità

- **Description:** gradi di libertà della devianza residua
- **Formula:**

$$n - k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> df.residual(object = modello)

[1] 23
```

hatvalues()

- **Package:** stats
- **Input:**

  `model` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> hatvalues(model = modello)

            1             2             3             4             5             6
1.105792e-01 9.362145e-02 5.003535e-02 7.003405e-02 5.631849e-02 5.828511e-02
            7             8             9            10            11            12
7.257287e-02 7.885661e-02 7.190461e-02 6.673601e-02 7.091234e-02 7.562508e-02
           13            14            15            16            17            18
6.946860e-02 7.717999e-02 8.349045e-02 1.068393e-01 1.054680e-01 1.211568e-01
           19            20            21            22            23            24
1.633692e-01 1.353446e-01 1.339136e-01 8.064188e-02 3.374658e-02 1.389985e-02
           25
4.030027e-15
```

cooks.distance()

- **Package:** stats
- **Input:**

  `model` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> cooks.distance(model = modello)
```

```
            1             2             3             4             5             6
3.938916e-01 4.483042e-01 1.454921e-01 1.984188e-01 1.430242e-01 7.411901e-02
            7             8             9            10            11            12
5.402610e-02 5.512482e-03 5.164813e-02 6.653361e-02 1.270601e-01 1.964540e-01
           13            14            15            16            17            18
1.127717e-02 2.016302e-01 5.316254e-01 8.928832e-02 1.540260e-02 2.994339e-01
           19            20            21            22            23            24
1.153996e-01 1.507299e-01 3.110377e-01 7.571077e-01 4.134756e-01 8.617915e-01
           25
4.693465e-28
```

cookd()

- **Package:** car
- **Input:**

  model modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cloglog"))
> cookd(model = modello)

            1             2             3             4             5             6
3.938916e-01 4.483042e-01 1.454921e-01 1.984188e-01 1.430242e-01 7.411901e-02
            7             8             9            10            11            12
5.402610e-02 5.512482e-03 5.164813e-02 6.653361e-02 1.270601e-01 1.964540e-01
           13            14            15            16            17            18
1.127717e-02 2.016302e-01 5.316254e-01 8.928832e-02 1.540260e-02 2.994339e-01
           19            20            21            22            23            24
1.153996e-01 1.507299e-01 3.110377e-01 7.571077e-01 4.134756e-01 8.617915e-01
           25
4.693465e-28
```

Capitolo 20

Regressione di Cauchy

20.1 Simbologia

$$F_U^{-1}(\pi_i) = \beta_1 + \beta_2\, x_{i1} + \beta_3\, x_{i2} + \cdots + \beta_k\, x_{ik-1} \qquad Y_i \sim \text{Bin}(\pi_i, n_i) \quad \forall\, i = 1, 2, \ldots, n \qquad U \sim \text{Cauchy}(0, 1)$$

- 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 : $X_i = (1, x_{i1}, x_{i2}, \ldots, x_{ik-1}) \quad \forall\, i = 1, 2, \ldots, n$
- vettore numerico positivo dei pesi IWLS: $w = (w_1, w_2, \ldots, w_n)$
- 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 $n \times n$: $H = X\,(X^T\, W^{-1}\, X)^{-1}\, X^T\, W^{-1}$
- valori di leva: $h_i = H_{i,i} \quad \forall\, i = 1, 2, \ldots, n$
- distanza di *Cook*: $cd_i = \left(e_i^P\right)^2 \frac{h_i}{k\,(1-h_i)^2} \quad \forall\, i = 1, 2, \ldots, n$
- stime IWLS: $\hat{\beta} = \left(\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_k\right)^T$
- standard error delle stime IWLS: $s_{\hat{\beta}} = \sqrt{\text{diag}((X^T\, W^{-1}\, X)^{-1})}$
- z-values delle stime IWLS: $z_{\hat{\beta}} = \hat{\beta} \,/\, s_{\hat{\beta}}$
- correlazione delle stime IWLS: $r_{\hat{\beta}_i\, \hat{\beta}_j} = \frac{(X^T\, W^{-1}\, X)^{-1}_{i,\,j}}{s_{\hat{\beta}_i}\, s_{\hat{\beta}_j}} \quad \forall\, i, j = 1, 2, \ldots, k$
- residui di devianza: $e_i = \text{sign}\,(y_i - \hat{y}_i) \sqrt{2 \left[y_i \log\left(\frac{y_i}{\hat{y}_i} + C_{i1}\right) + (n_i - y_i) \log\left(\frac{n_i - y_i}{n_i - \hat{y}_i} + C_{i2}\right)\right]}$

  $\forall\, i = 1, 2, \ldots, n$ dove $C_{i1} = 0.5\,(1 - \text{sign}(y_i)) \,/\, \hat{y}_i$ e $C_{i2} = 0.5\,(1 - \text{sign}(n_i - y_i)) \,/\, (n_i - \hat{y}_i)$
- residui standard: $rstandard_i = e_i \,/\, \sqrt{1 - h_i} \quad \forall\, i = 1, 2, \ldots, n$
- residui studentizzati: $rstudent_i = \text{sign}\,(y_i - \hat{y}_i) \sqrt{e_i^2 + h_i\, \left(e_i^P\right)^2 \,/\, (1 - h_i)} \quad \forall\, i = 1, 2, \ldots, n$
- residui di *Pearson*: $e_i^P = \frac{y_i - n_i\, \hat{\pi}_i}{\sqrt{n_i\, \hat{\pi}_i\, (1 - \hat{\pi}_i)}} \quad \forall i = 1, 2, \ldots, n$
- residui di lavoro: $e_i^W = \frac{y_i - n_i\, \hat{\pi}_i}{n_i\, \hat{\pi}_i\, (1 - \hat{\pi}_i)} \quad \forall i = 1, 2, \ldots, n$
- residui di riposta: $e_i^R = y_i - \hat{\mu}_i \quad \forall i = 1, 2, \ldots, n$
- log-verosimiglianza binomiale: $\hat{\ell} = \sum_{i=1}^n \left[\log\binom{n_i}{y_i} + y_i \log\left(\frac{\hat{y}_i}{n_i}\right) + (n_i - y_i) \log\left(1 - \frac{\hat{y}_i}{n_i}\right)\right]$
- valori adattati: $\hat{\pi}_i = F_U\left(X_i\, \hat{\beta}\right) \quad \forall\, i = 1, 2, \ldots, n$

- numero di successi attesi: $\hat{y}_i \, = \, n_i \, \hat{\pi}_i \quad \forall \, i \, = \, 1, \, 2, \, \ldots, \, n$
- log-verosimiglianza binomiale modello saturo: $\hat{\ell}_{saturo} \, = \, \sum_{i=1}^{n} \left[\log \binom{n_i}{y_i} + y_i \, \log \left(\frac{y_i}{n_i} \right) + (n_i - y_i) \, \log \left(1 - \frac{y_i}{n_i} \right) \right]$
- devianza residua: $D \, = \, 2 \, \left(\hat{\ell}_{saturo} - \hat{\ell} \right) \, = \, \sum_{i=1}^{n} \, e_i^2$
- gradi di libertà della devianza residua: $n - k$
- log-verosimiglianza binomiale modello nullo: $\hat{\ell}_{nullo} \, = \, \sum_{i=1}^{n} \left[\log \binom{n_i}{y_i} + y_i \, \log \left(\hat{\pi} \right) + (n_i - y_i) \, \log \left(1 - \hat{\pi} \right) \right]$
- valori adattati modello nullo: $\hat{\pi} \, = \, \sum_{j=1}^{n} y_j \, / \, \sum_{j=1}^{n} n_j \quad \forall \, i \, = \, 1, \, 2, \, \ldots, \, n$
- numero di successi attesi modello nullo: $\hat{y}_i \, = \, n_i \, \hat{\pi} \quad \forall \, i \, = \, 1, \, 2, \, \ldots, \, n$
- devianza residua modello nullo: $D_{nullo} \, = \, 2 \, \left(\hat{\ell}_{saturo} - \hat{\ell}_{nullo} \right)$
- gradi di libertà della devianza residua modello nullo: $n - 1$
- stima IWLS intercetta modello nullo: $\hat{\beta}_{nullo} \, = \, F_U^{-1} \left(\hat{\pi} \right)$

20.2 Stima

glm()

- **Package:** `stats`
- **Input:**

  `formula` modello di regressione di *Cauchy* con $k - 1$ variabili esplicative ed n unità

  `family = binomial(link="cauchit")` famiglia e link del modello

  `x = TRUE` matrice del modello

- **Description:** analisi di regressione di *Cauchy*
- **Output:**

  `coefficients` stime IWLS

  `residuals` residui di lavoro

  `fitted.values` valori adattati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` 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`

  $$\hat{\beta}_j \quad \forall \, j \, = \, 1, \, 2, \, \ldots, \, k$$

  `residuals`

  $$e_i^W \quad \forall \, i \, = \, 1, \, 2, \, \ldots, \, n$$

  `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

$$w_i \quad \forall\, i\, =\, 1,\, 2,\, \dots,\, n$$

prior.weights

$$n_i \quad \forall\, i\, =\, 1,\, 2,\, \dots,\, n$$

df.residual

$$n-k$$

df.null

$$n-1$$

y

$$y_i\,/\,n_i \quad \forall\, i\, =\, 1,\, 2,\, \dots,\, n$$

x

$$X$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"),
+     x = TRUE)
> modello$coefficients

(Intercept)           x
 -33.544126    2.583834

> modello$residuals

         1          2          3          4          5          6          7
-9.8152648 -7.2558854 -6.3140094 -4.0086223 -3.2932991 -0.9917917  0.4226277
         8          9         10         11         12         13         14
 1.5498952  0.6272238  1.7058520  0.9553468  0.3321975 -0.3474066 -0.5728429
        15         16         17         18         19         20         21
-0.4855652 -2.0313711 -2.4430322  0.6948164  0.9814772 -0.2170523  1.6310583
        22         23         24         25
 1.8963437  3.7327336  4.4091809 11.9357223

> modello$fitted.values
```

```
         1          2          3          4          5          6          7
0.03254332 0.04415163 0.05084422 0.05663242 0.06388783 0.07323785 0.08571643
         8          9         10         11         12         13         14
0.10314181 0.12897631 0.17045144 0.24383760 0.38066032 0.57870619 0.73297838
        15         16         17         18         19         20         21
0.81708886 0.86366984 0.89210300 0.91098535 0.92435062 0.93427641 0.94192536
        22         23         24         25
0.94799380 0.95292239 0.95700290 0.97326854
```

```
> modello$rank
```

```
[1] 2
```

```
> modello$linear.predictors
```

```
         1          2          3          4          5          6          7
-9.7470111 -7.1631766 -6.2071579 -5.5611993 -4.9152406 -4.2692820 -3.6233234
         8          9         10         11         12         13         14
-2.9773648 -2.3314062 -1.6854476 -1.0394890 -0.3935303  0.2524283  0.8983869
        15         16         17         18         19         20         21
 1.5443455  2.1903041  2.8362627  3.4822213  4.1281800  4.7741386  5.4200972
        22         23         24         25
 6.0660558  6.7120144  7.3579730 11.8796833
```

```
> modello$deviance
```

```
[1] 180.8584
```

```
> modello$aic
```

```
[1] 268.9102
```

```
> modello$null.deviance
```

```
[1] 3693.884
```

```
> modello$weights
```

```
          1           2           3           4           5           6
 0.13128604  0.17547429  0.12496388  0.22326973  0.24087950  0.35536805
          7           8           9          10          11          12
 0.68009289  1.24943550  2.17782383  4.51791817 12.69591273 34.80291036
         13          14          15          16          17          18
36.35987656 16.80244939  6.21201298  2.99536877  1.26102284  0.70343728
         19          20          21          22          23          24
 0.53414690  0.29731270  0.24487355  0.15967458  0.10010712  0.09232367
         25
 0.20223732
```

```
> modello$prior.weights
```

```
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
 376  200   93  120   90   88  105  111  100   93  100  108   99  106  105  117
  17   18   19   20   21   22   23   24   25
  98   97  120  102  122  111   94  114 1049
```

```
> modello$df.residual
```

```
[1] 23
```

```
> modello$df.null

[1] 24

> modello$y

         1          2          3          4          5          6          7
0.00000000 0.00000000 0.00000000 0.01666667 0.02222222 0.05681818 0.09523810
         8          9         10         11         12         13         14
0.15315315 0.16000000 0.31182796 0.39000000 0.47222222 0.47474747 0.63207547
        15         16         17         18         19         20         21
0.77142857 0.75213675 0.80612245 0.92783505 0.94166667 0.93137255 0.95901639
        22         23         24         25
0.96396396 0.97872340 0.98245614 1.00000000

> modello$x

   (Intercept)     x
1            1  9.21
2            1 10.21
3            1 10.58
4            1 10.83
5            1 11.08
6            1 11.33
7            1 11.58
8            1 11.83
9            1 12.08
10           1 12.33
11           1 12.58
12           1 12.83
13           1 13.08
14           1 13.33
15           1 13.58
16           1 13.83
17           1 14.08
18           1 14.33
19           1 14.58
20           1 14.83
21           1 15.08
22           1 15.33
23           1 15.58
24           1 15.83
25           1 17.58
attr(,"assign")
[1] 0 1
```

summary.glm()

- **Package:** `stats`
- **Input:**

  `object` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

  `correlation = TRUE` correlazione delle stime IWLS

- **Description:** analisi di regressione di *Cauchy*
- **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 IWLS non scalata

`cov.scaled` matrice di covarianza delle stime IWLS scalata

`correlation` matrice di correlazione delle stime IWLS

- **Formula:**

`deviance`

$$D$$

`aic`

$$-2\,\hat{\ell} + 2\,k$$

`df.residual`

$$n - k$$

`null.deviance`

$$D_{nullo}$$

`df.null`

$$n - 1$$

`deviance.resid`

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

`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`

$$(X^T\,W^{-1}\,X)^{-1}$$

`correlation`

$$r_{\hat{\beta}_i\,\hat{\beta}_j} \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> res <- summary.glm(object = modello, correlation = TRUE)
> res$deviance

[1] 180.8584

> res$aic

[1] 268.9102

> res$df.residual

[1] 23
```

```
> res$null.deviance

[1] 3693.884

> res$df.null

[1] 24

> res$deviance.resid

         1          2          3          4          5          6          7
-4.9879493 -4.2499874 -3.1154320 -2.2134735 -1.8547635 -0.6138012  0.3429411
         8          9         10         11         12         13         14
 1.6292015  0.8969607  3.3340955  3.2290861  1.9359119 -2.0794099 -2.2707637
        15         16         17         18         19         20         21
-1.1752053 -3.2150141 -2.5014455  0.6008633  0.7452777 -0.1175573  0.8498527
        22         23         24         25
 0.8002034  1.3186785  1.5146367  7.5396162

> res$coefficients

              Estimate Std. Error   z value     Pr(>|z|)
(Intercept) -33.544126  2.1690507 -15.46489 5.987702e-54
x             2.583834  0.1668083  15.48984 4.063009e-54

> res$cov.unscaled

            (Intercept)           x
(Intercept)   4.7047808 -0.36150385
x            -0.3615038  0.02782502

> res$cov.scaled

            (Intercept)           x
(Intercept)   4.7047808 -0.36150385
x            -0.3615038  0.02782502

> res$correlation

            (Intercept)         x
(Intercept)    1.000000 -0.999138
x             -0.999138  1.000000
```

glm.fit()

- **Package:** `stats`
- **Input:**

  `x` matrice del modello

  `y` proporzione di successi

  `weights` numero di prove

  `family = binomial(link="cauchit")` famiglia e link del modello

- **Description:** analisi di regressione di *Cauchy*
- **Output:**

  `coefficients` stime IWLS

`residuals` residui di lavoro
`fitted.values` valori adattati
`rank` rango della matrice del modello
`linear.predictors` predittori lineari
`deviance` 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`
$$\hat{\beta}_j \quad \forall\, j = 1, 2, \dots, k$$
`residuals`
$$e_i^W \quad \forall\, i = 1, 2, \dots, n$$
`fitted.values`
$$\hat{\pi}_i \quad \forall\, i = 1, 2, \dots, n$$
`rank`
$$k$$
`linear.predictors`
$$X\,\hat{\beta}$$
`deviance`
$$D$$
`aic`
$$-2\,\hat{\ell} + 2\,k$$
`null.deviance`
$$D_{nullo}$$
`weights`
$$w_i \quad \forall\, i = 1, 2, \dots, n$$
`prior.weights`
$$n_i \quad \forall\, i = 1, 2, \dots, n$$
`df.residual`
$$n - k$$
`df.null`
$$n - 1$$
`y`
$$y_i \,/\, n_i \quad \forall\, i = 1, 2, \dots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "logit"))
> X <- model.matrix(object = modello)
> res <- glm.fit(x = X, y/Total, weights = Total, family = binomial(link = "logit"))
> res$coefficients
```

```
(Intercept)            x
 -21.226395    1.631968
```

```
> res$residuals
```

```
 [1] -1.00203763 -1.01042031 -1.01905988 -0.41336424 -0.48212701 -0.07089826
 [7]  0.07938086  0.22704866 -0.13926878  0.33629857  0.25835047  0.17881393
[13] -0.22141017  0.01336452  0.26283804 -0.24965088 -0.36552096  0.33713195
[19]  0.19514514 -0.43506531 -0.25760272 -0.64783388 -0.44626460 -0.78405425
[25]  1.00057358
```

```
> res$fitted.values
```

```
 [1] 0.002033490 0.010312851 0.018703394 0.027863526 0.041320994 0.060871141
 [7] 0.088814107 0.127838223 0.180610428 0.248949062 0.332647930 0.428434554
[13] 0.529902047 0.628956590 0.718237396 0.793102235 0.852169542 0.896572801
[19] 0.928753893 0.951463983 0.967190831 0.977939948 0.985221193 0.990123427
[25] 0.999426746
```

```
> res$rank
```

```
[1] 2
```

```
> res$linear.predictors
```

```
 [1] -6.1959664 -4.5639981 -3.9601698 -3.5521777 -3.1441856 -2.7361935
 [7] -2.3282014 -1.9202093 -1.5122173 -1.1042252 -0.6962331 -0.2882410
[13]  0.1197511  0.5277432  0.9357353  1.3437274  1.7517194  2.1597115
[19]  2.5677036  2.9756957  3.3836878  3.7916799  4.1996720  4.6076640
[25]  7.4636087
```

```
> res$deviance
```

```
[1] 26.70345
```

```
> res$aic
```

```
[1] 114.7553
```

```
> res$null.deviance
```

```
[1] 3693.884
```

```
> res$weights
```

```
 [1]  0.7630428  2.0413099  1.7068902  3.2504707  3.5652333  5.0306085
 [7]  8.4972661 12.3760338 14.7990471 17.3885402 22.1993347 26.4468672
[13] 24.6614810 24.7372446 21.2491158 19.1986735 12.3457255  8.9948289
[19]  7.9404319  4.7104022  3.8714069  2.3946581  1.3686835  1.1148148
[25]  0.6010036
```

```
> res$prior.weights
```

```
 [1]  376  200   93  120   90   88  105  111  100   93  100  108   99  106  105
[16]  117   98   97  120  102  122  111   94  114 1049
```

```
> res$df.residual
```

```
[1] 23
```

```
> res$df.null
```

```
[1] 24
```

```
> res$y
```

```
 [1] 0.00000000 0.00000000 0.00000000 0.01666667 0.02222222 0.05681818
 [7] 0.09523810 0.15315315 0.16000000 0.31182796 0.39000000 0.47222222
[13] 0.47474747 0.63207547 0.77142857 0.75213675 0.80612245 0.92783505
[19] 0.94166667 0.93137255 0.95901639 0.96396396 0.97872340 0.98245614
[25] 1.00000000
```

vcov()

- **Package:** stats
- **Input:**

  object modello di regressione di *Cauchy* con $k - 1$ variabili esplicative ed n unità

- **Description:** matrice di covarianza delle stime IWLS
- **Formula:**

$$(X^T W^{-1} X)^{-1}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> vcov(object = modello)
```

```
            (Intercept)           x
(Intercept)   4.7047808 -0.36150385
x            -0.3615038  0.02782502
```

coef()

- **Package:** stats
- **Input:**

  object modello di regressione di *Cauchy* con $k - 1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall j = 1, 2, \dots, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> coef(object = modello)

(Intercept)          x
 -33.544126    2.583834
```

coefficients()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Cauchy* con $k - 1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> coefficients(object = modello)

(Intercept)          x
 -33.544126    2.583834
```

predict.glm()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Cauchy* con $k - 1$ variabili esplicative ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` standard error delle stime

- **Description:** previsione
- **Output:**

  `fit` valore previsto

  `se.fit` standard error delle stime

- **Formula:**

`fit`

$$x_0^T\,\hat{\beta}$$

`se.fit`

$$\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> res <- predict.glm(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit

        1
-30.18514

> res$se.fit

[1] 1.952408
```

predict()

- **Package:** `stats`
- **Input:**

  `object` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` standard error delle stime

- **Description:** previsione
- **Output:**

  `fit` valore previsto

  `se.fit` standard error delle stime

- **Formula:**

`fit`

$$x_0^T\,\hat{\beta}$$

`se.fit`

$$\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> res <- predict(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit
```

```
        1
-30.18514
```

```
> res$se.fit
```

```
[1] 1.952408
```

fitted()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità
- **Description:** valori adattati
- **Formula:**

$$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> fitted(object = modello)
```

```
         1          2          3          4          5          6          7
0.03254332 0.04415163 0.05084422 0.05663242 0.06388783 0.07323785 0.08571643
         8          9         10         11         12         13         14
0.10314181 0.12897631 0.17045144 0.24383760 0.38066032 0.57870619 0.73297838
        15         16         17         18         19         20         21
0.81708886 0.86366984 0.89210300 0.91098535 0.92435062 0.93427641 0.94192536
        22         23         24         25
0.94799380 0.95292239 0.95700290 0.97326854
```

fitted.values()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità
- **Description:** valori adattati
- **Formula:**

$$\hat{\pi}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> fitted.values(object = modello)
```

```
         1          2          3          4          5          6          7
0.03254332 0.04415163 0.05084422 0.05663242 0.06388783 0.07323785 0.08571643
         8          9         10         11         12         13         14
0.10314181 0.12897631 0.17045144 0.24383760 0.38066032 0.57870619 0.73297838
        15         16         17         18         19         20         21
0.81708886 0.86366984 0.89210300 0.91098535 0.92435062 0.93427641 0.94192536
        22         23         24         25
0.94799380 0.95292239 0.95700290 0.97326854
```

cov2cor()

- **Package:** stats
- **Input:**

  V matrice di covarianza delle stime IWLS di dimensione $k \times k$

- **Description:** converte la matrice di covarianza nella matrice di correlazione
- **Formula:**

$$r_{\hat{\beta}_i \, \hat{\beta}_j} \quad \forall \, i, j \, = \, 1, \, 2, \, \ldots, \, k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> V <- vcov(object = modello)
> cov2cor(V)
```

```
            (Intercept)         x
(Intercept)    1.000000 -0.999138
x             -0.999138  1.000000
```

20.3 Adattamento

logLik()

- **Package:** stats
- **Input:**

  object modello di regressione di *Cauchy* con $k - 1$ variabili esplicative ed n unità

- **Description:** log-verosimiglianza binomiale
- **Formula:**

$$\hat{\ell}$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> logLik(object = modello)

'log Lik.' -132.4551 (df=2)
```

AIC()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> AIC(object = modello)

[1] 268.9102
```

durbin.watson()

- **Package:** car
- **Input:**

  `model` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

  `dw` valore empirico della statistica *D–W*

- **Formula:**

`dw`

$$\sum_{i=2}^{n} (e_i - e_{i-1})^2 \,/\, D$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> durbin.watson(model = modello)
```

```
 lag Autocorrelation D-W Statistic p-value
   1       0.5390491     0.4700264       0
 Alternative hypothesis: rho != 0
```

```
> res <- durbin.watson(model = modello)
> res$dw
```

```
[1] 0.4700264
```

extractAIC()

- **Package:** `stats`
- **Input:**

  `fit` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** numero di parametri del modello ed indice *AIC* generalizzato
- **Formula:**

$$k \qquad -2\,\hat{\ell} + 2\,k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> extractAIC(fit = modello)
```

```
[1]   2.0000 268.9102
```

deviance()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** devianza residua
- **Formula:**

$$D$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> deviance(object = modello)

[1] 180.8584
```

anova()

- **Package:** stats
- **Input:**

  `nullo` modello nullo di regressione di *Cauchy* con n unità

  `modello` modello di regressione di *Cauchy* con $k-1$ variabili esplicative con n unità

  `test = "Chisq"`

- **Description:** 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`

$$df = k-1$$

  `Deviance`

$$c = D_{nullo} - D$$

  `P(>|Chi|)`

$$P(\chi^2_{df} \geq c)$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+      12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+      14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+      88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+      108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+      1049)
> nullo <- glm(formula = cbind(y, Total - y) ~ 1, family = binomial(link = "cauchit"))
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> anova(nullo, modello, test = "Chisq")

Analysis of Deviance Table

Model 1: cbind(y, Total - y) ~ 1
Model 2: cbind(y, Total - y) ~ x
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1        24     3693.9
2        23      180.9  1   3513.0       0.0

> res <- anova(nullo, modello, test = "Chisq")
> res$"Resid. Df"

[1] 24 23

> res$"Resid. Dev"

[1] 3693.8836  180.8584

> res$Df

[1] NA  1

> res$Deviance

[1]       NA 3513.025

> res$"P(>|Chi|)"

[1] NA  0
```

drop1()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Cauchy* con $k - 1$ variabili esplicative ed n unità

  `test = "Chisq"`
- **Description:** submodels
- **Output:**

  `Df` differenza tra gradi di libertà

  `Deviance` differenza tra devianze residue

  `AIC` indice AIC

  `LRT` valore empirico della statistica χ^2

  `Pr(Chi)` p-value

- **Formula:**

  `Df`

$$\underbrace{1,\,1,\,\ldots,\,1}_{k-1\text{ volte}}$$

  `Deviance`

$$D,\, D_{-x_j} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  dove D_{-x_j} rappresenta la devianza residua del modello eliminata la variabile esplicativa x_j.

  `AIC`

$$-2\,\hat{\ell} + 2\,k,\; -2\,\hat{\ell}_{-x_j} + 2\,(k-1) \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  dove $\hat{\ell}_{-x_j}$ rappresenta la log-verosimiglianza binomiale del modello eliminata la variabile esplicativa x_j.

  `LRT`

$$c_j \,=\, D_{-x_j} - D \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  `Pr(Chi)`

$$P(\chi^2_1 \geq c_j) \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> drop1(object = modello, test = "Chisq")

Single term deletions

Model:
cbind(y, Total - y) ~ x
       Df Deviance    AIC    LRT   Pr(Chi)
<none>       180.9  268.9
x       1   3693.9 3779.9 3513.0 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> res <- drop1(object = modello, test = "Chisq")
> res$Df

[1] NA  1

> res$Deviance

[1]  180.8584 3693.8836

> res$AIC

[1]  268.9102 3779.9354

> res$LRT

[1]       NA 3513.025

> res$"Pr(Chi)"

[1] NA  0
```

add1()

- **Package:** stats
- **Input:**

  `object` modello nullo di regressione di *Cauchy*

  `scope` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

  `test = "Chisq"`
- **Description:** submodels
- **Output:**

  `Df` differenza tra gradi di libertà

  `Deviance` differenza tra devianze residue

  `AIC` indice AIC

  `LRT` valore empirico della statistica χ^2

  `Pr(Chi)` p-value
- **Formula:**

  `Df`

$$\underbrace{1,\, 1,\, \ldots,\, 1}_{k-1 \text{ volte}}$$

  `Deviance`

$$D_{nullo},\, D_{x_j} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  dove D_{x_j} rappresenta la devianza residua del modello con la sola variabile esplicativa x_j.

  `AIC`

$$-2\,\hat{\ell}_{nullo} + 2,\; -2\,\hat{\ell}_{x_j} + 4 \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  dove $\hat{\ell}_{x_j}$ rappresenta la log-verosimiglianza binomiale del modello con la sola variabile esplicativa x_j.

  `LRT`

$$c_j \,=\, D_{nullo} - D_{x_j} \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

  `Pr(Chi)`

$$P(\chi^2_1 \geq c_j) \quad \forall\, j \,=\, 1,\, 2,\, \ldots,\, k-1$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> nullo <- glm(formula = cbind(y, Total - y) ~ 1, family = binomial(link = "cauchit"))
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> add1(object = nullo, scope = modello, test = "Chisq")

Single term additions

Model:
cbind(y, Total - y) ~ 1
       Df Deviance    AIC    LRT   Pr(Chi)
<none>      3693.9 3779.9
x       1    180.9  268.9 3513.0 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
> res <- add1(object = nullo, scope = modello, test = "Chisq")
> res$Df

[1] NA  1

> res$Deviance

[1] 3693.8836  180.8584

> res$AIC

[1] 3779.9354  268.9102

> res$LRT

[1]       NA 3513.025

> res$"Pr(Chi)"

[1] NA  0
```

20.4 Diagnostica

rstandard()

- **Package:** stats
- **Input:**

  `model` modello di regressione di *Cauchy* con $k - 1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> rstandard(model = modello)

         1          2          3          4          5          6          7
-5.1264853 -4.3358475 -3.1490590 -2.2484272 -1.8797967 -0.6232837  0.3506059
         8          9         10         11         12         13         14
 1.6777851  0.9291382  3.4984066  3.5293420  2.3265176 -2.4900358 -2.5224910
        15         16         17         18         19         20         21
-1.2457978 -3.3570127 -2.5688041  0.6134906  0.7613634 -0.1193833  0.8636473
        22         23         24         25
 0.8106387  1.3317047  1.5311383  8.0376682
```

rstandard.glm()

- **Package:** stats
- **Input:**

  model modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> rstandard.glm(model = modello)

         1          2          3          4          5          6          7
-5.1264853 -4.3358475 -3.1490590 -2.2484272 -1.8797967 -0.6232837  0.3506059
         8          9         10         11         12         13         14
 1.6777851  0.9291382  3.4984066  3.5293420  2.3265176 -2.4900358 -2.5224910
        15         16         17         18         19         20         21
-1.2457978 -3.3570127 -2.5688041  0.6134906  0.7613634 -0.1193833  0.8636473
        22         23         24         25
 0.8106387  1.3317047  1.5311383  8.0376682
```

rstudent()

- **Package:** stats
- **Input:**

  model modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> rstudent(model = modello)
```

```
        1         2         3         4         5         6         7
-5.0588500 -4.2941160 -3.1327370 -2.2391220 -1.8738045 -0.6226038  0.3508547
        8         9        10        11        12        13        14
 1.6840319  0.9311874  3.5275840  3.5611698  2.3353549 -2.4956524 -2.5390300
       15        16        17        18        19        20        21
-1.2499439 -3.3841296 -2.5822550  0.6127486  0.7601912 -0.1194079  0.8623051
       22        23        24        25
 0.8095676  1.3291375  1.5275625  7.7960241
```

rstudent.glm()

- **Package:** stats
- **Input:**

  `model` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> rstudent.glm(model = modello)
```

```
        1         2         3         4         5         6         7
-5.0588500 -4.2941160 -3.1327370 -2.2391220 -1.8738045 -0.6226038  0.3508547
        8         9        10        11        12        13        14
 1.6840319  0.9311874  3.5275840  3.5611698  2.3353549 -2.4956524 -2.5390300
       15        16        17        18        19        20        21
-1.2499439 -3.3841296 -2.5822550  0.6127486  0.7601912 -0.1194079  0.8623051
       22        23        24        25
 0.8095676  1.3291375  1.5275625  7.7960241
```

residuals.default()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** residui di lavoro
- **Formula:**

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> residuals.default(object = modello)

         1          2          3          4          5          6          7
-9.8152648 -7.2558854 -6.3140094 -4.0086223 -3.2932991 -0.9917917  0.4226277
         8          9         10         11         12         13         14
 1.5498952  0.6272238  1.7058520  0.9553468  0.3321975 -0.3474066 -0.5728429
        15         16         17         18         19         20         21
-0.4855652 -2.0313711 -2.4430322  0.6948164  0.9814772 -0.2170523  1.6310583
        22         23         24         25
 1.8963437  3.7327336  4.4091809 11.9357223
```

residuals()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo
- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> residuals(object = modello, type = "deviance")
```

```
         1          2          3          4          5          6          7
-4.9879493 -4.2499874 -3.1154320 -2.2134735 -1.8547635 -0.6138012  0.3429411
         8          9         10         11         12         13         14
 1.6292015  0.8969607  3.3340955  3.2290861  1.9359119 -2.0794099 -2.2707637
        15         16         17         18         19         20         21
-1.1752053 -3.2150141 -2.5014455  0.6008633  0.7452777 -0.1175573  0.8498527
        22         23         24         25
 0.8002034  1.3186785  1.5146367  7.5396162
```

- **Example 2:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> residuals(object = modello, type = "pearson")
```

```
         1          2          3          4          5          6          7
-3.5563874 -3.0394419 -2.2319966 -1.8941117 -1.6163149 -0.5912262  0.3485259
         8          9         10         11         12         13         14
 1.7324103  0.9256002  3.6257473  3.4039079  1.9597174 -2.0948691 -2.3482148
        15         16         17         18         19         20         21
-1.2102597 -3.5158214 -2.7434754  0.5827626  0.7173290 -0.1183527  0.8071359
        22         23         24         25
 0.7577756  1.1810403  1.3397363  5.3676317
```

- **Example 3:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> residuals(object = modello, type = "working")
```

```
         1          2          3          4          5          6          7
-9.8152648 -7.2558854 -6.3140094 -4.0086223 -3.2932991 -0.9917917  0.4226277
         8          9         10         11         12         13         14
 1.5498952  0.6272238  1.7058520  0.9553468  0.3321975 -0.3474066 -0.5728429
        15         16         17         18         19         20         21
-0.4855652 -2.0313711 -2.4430322  0.6948164  0.9814772 -0.2170523  1.6310583
        22         23         24         25
 1.8963437  3.7327336  4.4091809 11.9357223
```

- **Example 4:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> residuals(object = modello, type = "response")
```

```
          1           2           3           4           5           6
-0.032543316 -0.044151625 -0.050844224 -0.039965753 -0.041665609 -0.016419665
          7           8           9          10          11          12
 0.009521665  0.050011345  0.031023688  0.141376522  0.146162404  0.091561906
         13          14          15          16          17          18
-0.103958715 -0.100902908 -0.045660287 -0.111533087 -0.085980550  0.016849703
         19          20          21          22          23          24
 0.017316049 -0.002903864  0.017091031  0.015970168  0.025801013  0.025453243
         25
 0.026731456
```

residuals.glm()

- **Package:** stats
- **Input:**

  object modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

  type = "deviance" / "pearson" / "working" / "response" tipo di residuo

- **Description:** residui
- **Formula:**

type = "deviance"

$$e_i \quad \forall i = 1, 2, \ldots, n$$

type = "pearson"

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

type = "working"

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

type = "response"

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> residuals.glm(object = modello, type = "deviance")

         1          2          3          4          5          6          7
-4.9879493 -4.2499874 -3.1154320 -2.2134735 -1.8547635 -0.6138012  0.3429411
         8          9         10         11         12         13         14
 1.6292015  0.8969607  3.3340955  3.2290861  1.9359119 -2.0794099 -2.2707637
        15         16         17         18         19         20         21
-1.1752053 -3.2150141 -2.5014455  0.6008633  0.7452777 -0.1175573  0.8498527
        22         23         24         25
 0.8002034  1.3186785  1.5146367  7.5396162
```

- **Example 2:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> residuals.glm(object = modello, type = "pearson")

         1          2          3          4          5          6          7
-3.5563874 -3.0394419 -2.2319966 -1.8941117 -1.6163149 -0.5912262  0.3485259
         8          9         10         11         12         13         14
 1.7324103  0.9256002  3.6257473  3.4039079  1.9597174 -2.0948691 -2.3482148
        15         16         17         18         19         20         21
-1.2102597 -3.5158214 -2.7434754  0.5827626  0.7173290 -0.1183527  0.8071359
        22         23         24         25
 0.7577756  1.1810403  1.3397363  5.3676317
```

- **Example 3:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> residuals.glm(object = modello, type = "working")

         1          2          3          4          5          6          7
-9.8152648 -7.2558854 -6.3140094 -4.0086223 -3.2932991 -0.9917917  0.4226277
         8          9         10         11         12         13         14
 1.5498952  0.6272238  1.7058520  0.9553468  0.3321975 -0.3474066 -0.5728429
        15         16         17         18         19         20         21
-0.4855652 -2.0313711 -2.4430322  0.6948164  0.9814772 -0.2170523  1.6310583
        22         23         24         25
 1.8963437  3.7327336  4.4091809 11.9357223
```

- **Example 4:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> residuals.glm(object = modello, type = "response")

           1            2            3            4            5            6
-0.032543316 -0.044151625 -0.050844224 -0.039965753 -0.041665609 -0.016419665
           7            8            9           10           11           12
 0.009521665  0.050011345  0.031023688  0.141376522  0.146162404  0.091561906
          13           14           15           16           17           18
-0.103958715 -0.100902908 -0.045660287 -0.111533087 -0.085980550  0.016849703
          19           20           21           22           23           24
```

```
0.017316049 -0.002903864  0.017091031  0.015970168  0.025801013  0.025453243
         25
0.026731456
```

resid()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Cauchy* con $k - 1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo

- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> resid(object = modello, type = "deviance")
```

```
         1          2          3          4          5          6          7
-4.9879493 -4.2499874 -3.1154320 -2.2134735 -1.8547635 -0.6138012  0.3429411
         8          9         10         11         12         13         14
 1.6292015  0.8969607  3.3340955  3.2290861  1.9359119 -2.0794099 -2.2707637
        15         16         17         18         19         20         21
-1.1752053 -3.2150141 -2.5014455  0.6008633  0.7452777 -0.1175573  0.8498527
        22         23         24         25
 0.8002034  1.3186785  1.5146367  7.5396162
```

- **Example 2:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> resid(object = modello, type = "pearson")

         1          2          3          4          5          6          7
-3.5563874 -3.0394419 -2.2319966 -1.8941117 -1.6163149 -0.5912262  0.3485259
         8          9         10         11         12         13         14
 1.7324103  0.9256002  3.6257473  3.4039079  1.9597174 -2.0948691 -2.3482148
        15         16         17         18         19         20         21
-1.2102597 -3.5158214 -2.7434754  0.5827626  0.7173290 -0.1183527  0.8071359
        22         23         24         25
 0.7577756  1.1810403  1.3397363  5.3676317
```

- **Example 3:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> resid(object = modello, type = "working")

         1          2          3          4          5          6          7
-9.8152648 -7.2558854 -6.3140094 -4.0086223 -3.2932991 -0.9917917  0.4226277
         8          9         10         11         12         13         14
 1.5498952  0.6272238  1.7058520  0.9553468  0.3321975 -0.3474066 -0.5728429
        15         16         17         18         19         20         21
-0.4855652 -2.0313711 -2.4430322  0.6948164  0.9814772 -0.2170523  1.6310583
        22         23         24         25
 1.8963437  3.7327336  4.4091809 11.9357223
```

- **Example 4:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> resid(object = modello, type = "response")

           1            2            3            4            5            6
-0.032543316 -0.044151625 -0.050844224 -0.039965753 -0.041665609 -0.016419665
           7            8            9           10           11           12
 0.009521665  0.050011345  0.031023688  0.141376522  0.146162404  0.091561906
          13           14           15           16           17           18
-0.103958715 -0.100902908 -0.045660287 -0.111533087 -0.085980550  0.016849703
          19           20           21           22           23           24
 0.017316049 -0.002903864  0.017091031  0.015970168  0.025801013  0.025453243
          25
 0.026731456
```

weighted.residuals()

- **Package:** stats
- **Input:**

  `obj` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** residui pesati
- **Formula:**

$$e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> weighted.residuals(obj = modello)

         1          2          3          4          5          6          7
-4.9879493 -4.2499874 -3.1154320 -2.2134735 -1.8547635 -0.6138012  0.3429411
         8          9         10         11         12         13         14
 1.6292015  0.8969607  3.3340955  3.2290861  1.9359119 -2.0794099 -2.2707637
        15         16         17         18         19         20         21
-1.1752053 -3.2150141 -2.5014455  0.6008633  0.7452777 -0.1175573  0.8498527
        22         23         24         25
 0.8002034  1.3186785  1.5146367  7.5396162
```

weights()

- **Package:** stats
- **Input:**

  `object` modello di regressione log-log complementare con $k-1$ variabili esplicative ed n unità

- **Description:** pesi iniziali
- **Formula:**

$$n_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> weights(object = modello)
```

```
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
 376  200   93  120   90   88  105  111  100   93  100  108   99  106  105  117
  17   18   19   20   21   22   23   24   25
  98   97  120  102  122  111   94  114 1049
```

df.residual()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** gradi di libertà della devianza residua
- **Formula:**

$$n - k$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> df.residual(object = modello)
```

```
[1] 23
```

hatvalues()

- **Package:** stats
- **Input:**

  `model` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> hatvalues(model = modello)
```

```
         1          2          3          4          5          6          7 
0.05331688 0.03921264 0.02124288 0.03084999 0.02645658 0.03019599 0.04324501 
         8          9         10         11         12         13         14 
0.05707539 0.06806370 0.09172888 0.16291078 0.30759773 0.30262070 0.18962759 
        15         16         17         18         19         20         21 
0.11011800 0.08280894 0.05175594 0.04074176 0.04180850 0.03035654 0.03168976 
        22         23         24         25 
0.02557996 0.01946748 0.02143853 0.12008984 
```

cooks.distance()

- **Package:**
- **Input:**

  `model` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> cooks.distance(model = modello)

           1            2            3            4            5            6 
0.3762214804 0.1962136349 0.0552357880 0.0589188486 0.0364623856 0.0056112386 
           7            8            9           10           11           12 
0.0028692913 0.0963310836 0.0335706735 0.7308700108 1.3468893627 1.2320350055 
          13           14           15           16           17           18 
1.3653510505 0.7961188111 0.1018405155 0.6083887972 0.2166167590 0.0075183418 
          19           20           21           22           23           24 
0.0117156580 0.0002261279 0.0110091368 0.0077349710 0.0141216419 0.0200921981 
          25 
2.2344212321 
```

cookd()

- **Package:** car
- **Input:**

  `model` modello di regressione di *Cauchy* con $k-1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(9.21, 10.21, 10.58, 10.83, 11.08, 11.33, 11.58, 11.83,
+     12.08, 12.33, 12.58, 12.83, 13.08, 13.33, 13.58, 13.83, 14.08,
+     14.33, 14.58, 14.83, 15.08, 15.33, 15.58, 15.83, 17.58)
> y <- c(0, 0, 0, 2, 2, 5, 10, 17, 16, 29, 39, 51, 47, 67, 81,
+     88, 79, 90, 113, 95, 117, 107, 92, 112, 1049)
> Total <- c(376, 200, 93, 120, 90, 88, 105, 111, 100, 93, 100,
+     108, 99, 106, 105, 117, 98, 97, 120, 102, 122, 111, 94, 114,
+     1049)
> modello <- glm(formula = cbind(y, Total - y) ~ x, family = binomial(link = "cauchit"))
> cookd(model = modello)

           1            2            3            4            5            6
0.3762214804 0.1962136349 0.0552357880 0.0589188486 0.0364623856 0.0056112386
           7            8            9           10           11           12
0.0028692913 0.0963310836 0.0335706735 0.7308700108 1.3468893627 1.2320350055
          13           14           15           16           17           18
1.3653510505 0.7961188111 0.1018405155 0.6083887972 0.2166167590 0.0075183418
          19           20           21           22           23           24
0.0117156580 0.0002261279 0.0110091368 0.0077349710 0.0141216419 0.0200921981
          25
2.2344212321
```

Capitolo 21

Regressione di Poisson

21.1 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 $n \times k$: $\quad X$
- numero di parametri da stimare e rango della matrice del modello: $\quad k$
- numero di unità: $\quad n$
- i-esima riga della matrice del modello : $\quad X_i \,=\, (1,\, x_{i1},\, x_{i2}, \ldots,\, x_{ik-1}) \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- vettore numerico positivo dei pesi IWLS: $\quad w \,=\, (w_1,\, w_2,\, \ldots,\, w_n)$
- matrice diagonale dei pesi IWLS di dimensione $n \times n$: $\quad W \,=\, \text{diag}(w_1^{-1},\, w_2^{-1},\, \ldots,\, w_n^{-1})$
- matrice di proiezione di dimensione $n \times n$: $\quad H \,=\, X\,(X^T\,W^{-1}\,X)^{-1}\,X^T\,W^{-1}$
- valori di leva: $\quad h_i \,=\, H_{i,i} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- distanza di *Cook*: $\quad cd_i \,=\, \left(e_i^P\right)^2 \frac{h_i}{k\,(1-h_i)^2} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- stime IWLS: $\quad \hat{\beta} \,=\, \left(\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_k\right)^T$
- standard error delle stime IWLS: $\quad s_{\hat{\beta}} \,=\, \sqrt{\text{diag}((X^T\,W^{-1}\,X)^{-1})}$
- z-values delle stime IWLS: $\quad z_{\hat{\beta}} \,=\, \hat{\beta}\,/\,s_{\hat{\beta}}$
- correlazione delle stime IWLS: $\quad r_{\hat{\beta}_i\,\hat{\beta}_j} \,=\, \frac{(X^T\,W^{-1}\,X)^{-1}_{i,\,j}}{s_{\hat{\beta}_i}\,s_{\hat{\beta}_j}} \quad \forall\, i, j \,=\, 1,\, 2,\, \ldots,\, k$
- residui di devianza: $\quad e_i \,=\, \text{sign}\,(y_i - \hat{\mu}_i)\,\sqrt{2\,\left(y_i\,\log\left(\frac{y_i}{\hat{\mu}_i} + C_i\right) - (y_i - \hat{\mu}_i)\right)}$

  dove $\quad C_i \,=\, 0.5\,(1 - \text{sign}(y_i))\,/\,\hat{\mu}_i \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- residui standard: $\quad rstandard_i \,=\, e_i\,/\,\sqrt{1 - h_i} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- residui studentizzati: $\quad rstudent_i \,=\, \text{sign}\,(y_i - \hat{\mu}_i)\,\sqrt{e_i^2 + h_i\,\left(e_i^P\right)^2\,/\,(1 - h_i)} \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- residui di *Pearson*: $\quad e_i^P \,=\, (y_i - \hat{\mu}_i)\,/\,\sqrt{\hat{\mu}_i} \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$
- residui di lavoro: $\quad e_i^W \,=\, (y_i - \hat{\mu}_i)\,/\,\hat{\mu}_i \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$
- residui di riposta: $\quad e_i^R \,=\, y_i - \hat{\mu}_i \quad \forall i \,=\, 1,\, 2,\, \ldots,\, n$
- log-verosimiglianza di *Poisson*: $\quad \hat{\ell} \,=\, \sum_{i=1}^n\,[y_i\,\log(\hat{\mu}_i) - \hat{\mu}_i - \log(y_i!)]$
- valori adattati: $\quad \hat{\mu}_i \,=\, \exp\left(X_i\,\hat{\beta}\right) \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$
- log-verosimiglianza di *Poisson* modello saturo: $\quad \hat{\ell}_{saturo} \,=\, \sum_{i=1}^n\,[y_i\,\log(y_i) - y_i - \log(y_i!)]$

- devianza residua: $D = 2\left(\hat{\ell}_{saturo} - \hat{\ell}\right) = \sum_{i=1}^{n} e_i^2 = 2\sum_{i=1}^{n} y_i \log\left(\frac{y_i}{\hat{\mu}_i} + C_i\right)$

  dove $C_i = 0.5\,(1 - \text{sign}(y_i))\,/\,\hat{\mu}_i \quad \forall\, i = 1, 2, \ldots, n$

- gradi di libertà della devianza residua: $n - k$
- log-verosimiglianza di *Poisson* modello nullo: $\hat{\ell}_{nullo} = \sum_{i=1}^{n} \left[y_i \log\left(\bar{y}\right) - \bar{y} - \log(y_i!)\right]$
- valori adattati modello nullo: $\hat{\mu} = \bar{y} \quad \forall\, i = 1, 2, \ldots, n$
- devianza residua modello nullo: $D_{nullo} = 2\left(\hat{\ell}_{saturo} - \hat{\ell}_{nullo}\right)$
- gradi di libertà della devianza residua modello nullo: $n - 1$
- stima IWLS intercetta modello nullo: $\hat{\beta}_{nullo} = \log\left(\hat{\mu}\right)$

21.2 Stima

glm()

- **Package:** `stats`
- **Input:**

  `formula` modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità

  `family = poisson(link="log")` famiglia e link del modello

  `x = TRUE` matrice del modello

- **Description:** analisi di regressione di *Poisson*
- **Output:**

  `coefficients` stime IWLS

  `residuals` residui di lavoro

  `fitted.values` valori adattati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` 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`

  $$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

  `residuals`

  $$e_i^W \quad \forall\, i = 1, 2, \ldots, n$$

  `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`

$$X$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"),
+     x = TRUE)
> modello$coefficients

(Intercept)           x
0.916392046 0.001997418

> modello$residuals

          1           2           3           4           5           6
-0.20165148 -0.56413249  0.29042202  0.70199431  0.34247005 -0.43487568
          7           8           9          10          11          12
 0.16386402 -0.20455645  0.04999536  0.33172955 -0.33831611  0.32602805
         13          14          15          16          17          18
 0.87408986 -0.35912141  0.10943462 -0.40119990  0.08161077 -0.33034568
         19          20          21          22          23          24
 0.50898714  0.21924503 -0.15404144 -0.68653798 -0.03098119 -0.37430000
         25          26          27          28          29          30
-0.17573412  1.66878447  0.56630428 -0.10405228  0.04163966 -0.71290188
         31          32
-0.46243717 -0.65221412

> modello$fitted.values

        1         2         3         4         5         6         7         8
 7.515515  9.177101 13.173985  5.287914 10.428538 14.156177  4.296035  8.800122
        9        10        11        12        13        14        15        16
 6.666696  5.256322  9.067774  6.033055 14.940586  6.241432  9.013600  6.680026
       17        18        19        20        21        22        23        24
 7.396376 13.439770 15.242012  7.381617  7.092546  3.190179  9.287745  6.392840
       25        26        27        28        29        30        31        32
10.918807  5.245834 10.853574 11.161366  6.720174 10.449389 16.742229  5.750665
```

```
> modello$rank

[1] 2

> modello$linear.predictors

       1        2        3        4        5        6        7        8
2.016970 2.216711 2.578244 1.665424 2.344546 2.650151 1.457692 2.174766
       9       10       11       12       13       14       15       16
1.897124 1.659432 2.204727 1.797253 2.704081 1.831210 2.198735 1.899122
      17       18       19       20       21       22       23       24
2.000990 2.598218 2.724056 1.998993 1.959044 1.160077 2.228696 1.855179
      25       26       27       28       29       30       31       32
2.390487 1.657434 2.384494 2.412458 1.905114 2.346544 2.817934 1.749315

> modello$deviance

[1] 62.8054

> modello$aic

[1] 190.1035

> modello$null.deviance

[1] 103.7138

> modello$weights

        1         2         3         4         5         6         7         8
 7.515661  9.177255 13.174144  5.288041 10.428696 14.156336  4.296149  8.800275
        9        10        11        12        13        14        15        16
 6.666836  5.256449  9.067928  6.033189 14.940742  6.241568  9.013754  6.680166
       17        18        19        20        21        22        23        24
 7.396521 13.439929 15.242168  7.381762  7.092689  3.190277  9.287900  6.392978
       25        26        27        28        29        30        31        32
10.918966  5.245960 10.853733 11.161525  6.720315 10.449547 16.742380  5.750797

> modello$prior.weights

 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
27 28 29 30 31 32
 1  1  1  1  1  1

> modello$df.residual

[1] 30

> modello$df.null

[1] 31

> modello$y

 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
 6  4 17  9 14  8  5  7  7  7  6  8 28  4 10  4  8  9 23  9  6  1  9  4  9 14
27 28 29 30 31 32
17 10  7  3  9  2
```

```
> modello$x
```

```
   (Intercept)   x
1            1 551
2            1 651
3            1 832
4            1 375
5            1 715
6            1 868
7            1 271
8            1 630
9            1 491
10           1 372
11           1 645
12           1 441
13           1 895
14           1 458
15           1 642
16           1 492
17           1 543
18           1 842
19           1 905
20           1 542
21           1 522
22           1 122
23           1 657
24           1 470
25           1 738
26           1 371
27           1 735
28           1 749
29           1 495
30           1 716
31           1 952
32           1 417
attr(,"assign")
[1] 0 1
```

summary.glm()

- **Package:** stats
- **Input:**

  object modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità

  correlation = TRUE correlazione delle stime IWLS

- **Description:** 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 IWLS non scalata

  cov.scaled matrice di covarianza delle stime IWLS scalata `correlation` matrice di correlazione delle stime IWLS

- **Formula:**

`deviance`

$$D$$

`aic`

$$-2\,\hat{\ell} + 2\,k$$

`df.residual`

$$n - k$$

`null.deviance`

$$D_{nullo}$$

`df.null`

$$n - 1$$

`deviance.resid`

$$e_i \quad \forall\, i = 1, 2, \ldots, n$$

`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`

$$(X^T\,W^{-1}\,X)^{-1}$$

`correlation`

$$r_{\hat{\beta}_i\,\hat{\beta}_j} \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> res <- summary.glm(object = modello, correlation = TRUE)
> res$deviance

[1] 62.8054

> res$aic

[1] 190.1035

> res$df.residual

[1] 30

> res$null.deviance

[1] 103.7138

> res$df.null

[1] 31

> res$deviance.resid
```

```
         1          2          3          4          5          6          7
-0.5731569 -1.9263607  1.0084275  1.4656879  1.0504241 -1.7835363  0.3309445
         8          9         10         11         12         13         14
-0.6294980  0.1280339  0.7234253 -1.0862504  0.7623113  3.0093299 -0.9610107
        15         16         17         18         19         20         21
 0.3228171 -1.1213526  0.2190303 -1.2890517  1.8466732  0.5756799 -0.4215129
        22         23         24         25         26         27         28
-1.4353411 -0.0949116 -1.0171558 -0.5990789  3.1586571  1.7215083 -0.3539304
        29         30         31         32
 0.1072073 -2.7223502 -2.0764597 -1.8101537
```

```
> res$coefficients
```

```
               Estimate   Std. Error  z value     Pr(>|z|)
(Intercept) 0.916392046 0.2215541099 4.136200 3.531049e-05
x           0.001997418 0.0003184551 6.272213 3.559532e-10
```

```
> res$cov.unscaled
```

```
              (Intercept)             x
(Intercept)  4.908622e-02 -6.797742e-05
x           -6.797742e-05  1.014137e-07
```

```
> res$cov.scaled
```

```
              (Intercept)             x
(Intercept)  4.908622e-02 -6.797742e-05
x           -6.797742e-05  1.014137e-07
```

```
> res$correlation
```

```
            (Intercept)          x
(Intercept)   1.0000000 -0.9634665
x            -0.9634665  1.0000000
```

glm.fit()

- **Package:** `stats`
- **Input:**

  `x` matrice del modello

  `y` numero di conteggi

  `family = poisson(link="log")` famiglia e link del modello

- **Description:** analisi di regressione di *Poisson*
- **Output:**

  `coefficients` stime IWLS

  `residuals` residui di lavoro

  `fitted.values` valori adattati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` 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`

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

`residuals`

$$e_i^W \quad \forall\, i = 1, 2, \ldots, n$$

`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`

$$y_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> X <- model.matrix(object = modello)
> res <- glm.fit(x = X, y, family = poisson(link = "log"))
> res$coefficients

(Intercept)           x
0.916392046 0.001997418

> res$residuals

 [1] -0.20165148 -0.56413249  0.29042202  0.70199431  0.34247005 -0.43487568
 [7]  0.16386402 -0.20455645  0.04999536  0.33172955 -0.33831611  0.32602805
[13]  0.87408986 -0.35912141  0.10943462 -0.40119990  0.08161077 -0.33034568
[19]  0.50898714  0.21924503 -0.15404144 -0.68653798 -0.03098119 -0.37430000
[25] -0.17573412  1.66878447  0.56630428 -0.10405228  0.04163966 -0.71290188
[31] -0.46243717 -0.65221412
```

```
> res$fitted.values

 [1]  7.515515  9.177101 13.173985  5.287914 10.428538 14.156177  4.296035
 [8]  8.800122  6.666696  5.256322  9.067774  6.033055 14.940586  6.241432
[15]  9.013600  6.680026  7.396376 13.439770 15.242012  7.381617  7.092546
[22]  3.190179  9.287745  6.392840 10.918807  5.245834 10.853574 11.161366
[29]  6.720174 10.449389 16.742229  5.750665

> res$rank

[1] 2

> res$linear.predictors

 [1] 2.016970 2.216711 2.578244 1.665424 2.344546 2.650151 1.457692 2.174766
 [9] 1.897124 1.659432 2.204727 1.797253 2.704081 1.831210 2.198735 1.899122
[17] 2.000990 2.598218 2.724056 1.998993 1.959044 1.160077 2.228696 1.855179
[25] 2.390487 1.657434 2.384494 2.412458 1.905114 2.346544 2.817934 1.749315

> res$deviance

[1] 62.8054

> res$aic

[1] 190.1035

> res$null.deviance

[1] 103.7138

> res$weights

 [1]  7.515661  9.177255 13.174144  5.288041 10.428696 14.156336  4.296149
 [8]  8.800275  6.666836  5.256449  9.067928  6.033189 14.940742  6.241568
[15]  9.013754  6.680166  7.396521 13.439929 15.242168  7.381762  7.092689
[22]  3.190277  9.287900  6.392978 10.918966  5.245960 10.853733 11.161525
[29]  6.720315 10.449547 16.742380  5.750797

> res$prior.weights

 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

> res$df.residual

[1] 30

> res$df.null

[1] 31

> res$y

 [1]  6  4 17  9 14  8  5  7  7  7  6  8 28  4 10  4  8  9 23  9  6  1  9  4  9
[26] 14 17 10  7  3  9  2
```

vcov()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità

- **Description:** matrice di covarianza delle stime IWLS
- **Formula:**

$$(X^T W^{-1} X)^{-1}$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> vcov(object = modello)

              (Intercept)             x
(Intercept)  4.908622e-02 -6.797742e-05
x           -6.797742e-05  1.014137e-07
```

coef()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> coef(object = modello)

(Intercept)           x
0.916392046 0.001997418
```

coefficients()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> coefficients(object = modello)
```

```
(Intercept)           x
0.916392046 0.001997418
```

predict.glm()

- **Package:** stats
- **Input:**

  object modello di regressione gamma con $k-1$ variabili esplicative ed n unità

  newdata il valore di x_0

  se.fit = TRUE standard error delle stime

- **Description:** previsione
- **Output:**

  fit valore previsto

  se.fit standard error delle stime

- **Formula:**

  fit

$$x_0^T\,\hat{\beta}$$

  se.fit

$$\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> res <- predict.glm(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit
```

```
        1
0.9189887
```

```
> res$se.fit
```

```
[1] 0.2211553
```

predict()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` standard error delle stime
- **Description:** previsione
- **Output:**

  `fit` valore previsto

  `se.fit` standard error delle stime
- **Formula:**

  `fit`

$$x_0^T \, \hat{\beta}$$

  `se.fit`

$$\sqrt{x_0^T \, (X^T \, W^{-1} \, X)^{-1} \, x_0}$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> res <- predict(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit

        1
0.9189887

> res$se.fit

[1] 0.2211553
```

fitted()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità
- **Description:** valori adattati
- **Formula:**

$$\hat{\mu}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> fitted(object = modello)
```

```
        1         2         3         4         5         6         7         8
 7.515515  9.177101 13.173985  5.287914 10.428538 14.156177  4.296035  8.800122
        9        10        11        12        13        14        15        16
 6.666696  5.256322  9.067774  6.033055 14.940586  6.241432  9.013600  6.680026
       17        18        19        20        21        22        23        24
 7.396376 13.439770 15.242012  7.381617  7.092546  3.190179  9.287745  6.392840
       25        26        27        28        29        30        31        32
10.918807  5.245834 10.853574 11.161366  6.720174 10.449389 16.742229  5.750665
```

fitted.values()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{\mu}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> fitted.values(object = modello)

        1         2         3         4         5         6         7         8
 7.515515  9.177101 13.173985  5.287914 10.428538 14.156177  4.296035  8.800122
        9        10        11        12        13        14        15        16
 6.666696  5.256322  9.067774  6.033055 14.940586  6.241432  9.013600  6.680026
       17        18        19        20        21        22        23        24
 7.396376 13.439770 15.242012  7.381617  7.092546  3.190179  9.287745  6.392840
       25        26        27        28        29        30        31        32
10.918807  5.245834 10.853574 11.161366  6.720174 10.449389 16.742229  5.750665
```

cov2cor()

- **Package:** stats
- **Input:**

  `V` matrice di covarianza delle stime IWLS di dimensione $k \times k$

- **Description:** converte la matrice di covarianza nella matrice di correlazione
- **Formula:**

$$r_{\hat{\beta}_i\, \hat{\beta}_j} \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> V <- vcov(object = modello)
> cov2cor(V)
```

```
            (Intercept)          x
(Intercept)   1.0000000 -0.9634665
x            -0.9634665  1.0000000
```

21.3 Adattamento

logLik()

- **Package:** stats
- **Input:**

  object modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità

- **Description:** log-verosimiglianza di *Poisson*
- **Formula:**

$$\hat{\ell}$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> logLik(object = modello)
```

```
'log Lik.' -93.05175 (df=2)
```

AIC()

- **Package:** stats
- **Input:**

  object modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità

- **Description:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell} + 2\,k$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> AIC(object = modello)
```

```
[1] 190.1035
```

durbin.watson()

- **Package:** car
- **Input:**

  `model` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Description:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

  `dw` valore empirico della statistica *D–W*

- **Formula:**

  `dw`

$$\sum_{i=2}^{n} (e_i - e_{i-1})^2 \,/\, D$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> durbin.watson(model = modello)

 lag Autocorrelation D-W Statistic p-value
   1       0.1275698      1.687458   0.264
 Alternative hypothesis: rho != 0

> res <- durbin.watson(model = modello)
> res$dw

[1] 1.687458
```

extractAIC()

- **Package:** stats
- **Input:**

  `fit` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Description:** numero di parametri del modello ed indice *AIC* generalizzato
- **Formula:**

$$k \qquad -2\,\hat{\ell} + 2\,k$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> extractAIC(fit = modello)

[1]   2.0000 190.1035
```

deviance()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità
- **Description:** devianza residua
- **Formula:**

$$D$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> deviance(object = modello)

[1] 62.8054
```

anova()

- **Package:** stats
- **Input:**

  `nullo` modello nullo di regressione di *Poisson* con n unità

  `modello` modello di regressione di *Poisson* con $k-1$ variabili esplicative con n unità

  `test = "Chisq"`
- **Description:** 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`

$$df = k-1$$

  `Deviance`

$$c = D_{nullo} - D$$

  `P(>|Chi|)`

$$P(\chi^2_{df} \geq c)$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> nullo <- glm(formula = y ~ 1, family = poisson(link = "log"))
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> anova(nullo, modello, test = "Chisq")

Analysis of Deviance Table

Model 1: y ~ 1
Model 2: y ~ x
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1        31    103.714
2        30     62.805  1   40.908 1.595e-10

> res <- anova(nullo, modello, test = "Chisq")
> res$"Resid. Df"

[1] 31 30

> res$"Resid. Dev"

[1] 103.7138  62.8054

> res$Df

[1] NA  1

> res$Deviance

[1]       NA 40.90836

> res$"P(>|Chi|)"

[1]           NA 1.595374e-10
```

drop1()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

  `test = "Chisq"`
- **Description:** submodels
- **Output:**

  `Df` differenza tra gradi di libertà

  `Deviance` differenza tra devianze residue

  `AIC` indice AIC

  `LRT` valore empirico della statistica χ^2

  `Pr(Chi)` p-value
- **Formula:**

Df

$$\underbrace{1,\,1,\,\dots,\,1}_{k-1 \text{ volte}}$$

Deviance

$$D,\; D_{-x_j} \quad \forall\, j \,=\, 1,\, 2,\, \dots,\, k-1$$

dove D_{-x_j} rappresenta la devianza residua del modello eliminata la variabile esplicativa x_j.

AIC

$$-2\,\hat{\ell} + 2\,k,\; -2\,\hat{\ell}_{-x_j} + 2\,(k-1) \quad \forall\, j \,=\, 1,\, 2,\, \dots,\, k-1$$

dove $\hat{\ell}_{-x_j}$ rappresenta la log-verosimiglianza di *Poisson* del modello eliminata la variabile esplicativa x_j.

LRT

$$c_j \,=\, D_{-x_j} - D \quad \forall\, j \,=\, 1,\, 2,\, \dots,\, k-1$$

Pr(Chi)

$$P(\chi^2_1 \geq c_j) \quad \forall\, j \,=\, 1,\, 2,\, \dots,\, k-1$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> drop1(object = modello, test = "Chisq")

Single term deletions

Model:
y ~ x
       Df Deviance     AIC     LRT   Pr(Chi)
<none>      62.805 190.104
x       1  103.714 229.012  40.908 1.595e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> res <- drop1(object = modello, test = "Chisq")
> res$Df

[1] NA  1

> res$Deviance

[1]  62.8054 103.7138

> res$AIC

[1] 190.1035 229.0119

> res$LRT

[1]       NA 40.90836

> res$"Pr(Chi)"

[1]           NA 1.595374e-10
```

add1()

- **Package:** stats
- **Input:**

  object modello nullo di regressione di *Poisson*

  scope modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

  test = "Chisq"
- **Description:** submodels
- **Output:**

  Df differenza tra gradi di libertà

  Deviance differenza tra devianze residue

  AIC indice AIC

  LRT valore empirico della statistica χ^2

  Pr(Chi) p-value
- **Formula:**

  Df

  $$\underbrace{1,\,1,\,\ldots,\,1}_{k-1\text{ volte}}$$

  Deviance

  $$D_{nullo},\,D_{x_j}\quad\forall\,j\,=\,1,\,2,\,\ldots,\,k-1$$

  dove D_{x_j} rappresenta la devianza residua del modello con la sola variabile esplicativa x_j.

  AIC

  $$-2\,\hat{\ell}_{nullo}+2,\;-2\,\hat{\ell}_{x_j}+4\quad\forall\,j\,=\,1,\,2,\,\ldots,\,k-1$$

  dove $\hat{\ell}_{x_j}$ rappresenta la log-verosimiglianza di *Poisson* del modello con la sola variabile esplicativa x_j.

  LRT

  $$c_j\,=\,D_{nullo}-D_{x_j}\quad\forall\,j\,=\,1,\,2,\,\ldots,\,k-1$$

  Pr(Chi)

  $$P(\chi^2_1\geq c_j)\quad\forall\,j\,=\,1,\,2,\,\ldots,\,k-1$$
- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> nullo <- glm(formula = y ~ 1, family = poisson(link = "log"))
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> add1(object = nullo, scope = modello, test = "Chisq")

Single term additions

Model:
y ~ 1
       Df Deviance     AIC     LRT   Pr(Chi)
<none>     103.714 229.012
x       1   62.805 190.104  40.908 1.595e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> res <- add1(object = nullo, scope = modello, test = "Chisq")
> res$Df
```

```
[1] NA  1
> res$Deviance
[1] 103.7138  62.8054
> res$AIC
[1] 229.0119 190.1035
> res$LRT
[1]       NA 40.90836
> res$"Pr(Chi)"
[1]           NA 1.595374e-10
```

21.4 Diagnostica

rstandard()

- **Package:** stats
- **Input:**

  model modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> rstandard(model = modello)

          1           2           3           4           5           6
-0.58415822 -1.95861072  1.05211402  1.51608947  1.07143385 -1.88626732
          7           8           9          10          11          12
 0.34589794 -0.63996238  0.13103010  0.74852597 -1.10435414  0.78352354
         13          14          15          16          17          18
 3.22469291 -0.98623876  0.32818923 -1.14750260  0.22333743 -1.34944537
         19          20          21          22          23          24
 1.98995067  0.58703566 -0.43038260 -1.52017691 -0.09651101 -1.04276847
         25          26          27          28          29          30
-0.61255699  3.26857905  1.75959764 -0.36242210  0.10968144 -2.77705113
         31          32
-2.31245034 -1.86471908
```

rstandard.glm()

- **Package:** stats
- **Input:**

  `model` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> rstandard.glm(model = modello)

          1           2           3           4           5           6 
-0.58415822 -1.95861072  1.05211402  1.51608947  1.07143385 -1.88626732 
          7           8           9          10          11          12 
 0.34589794 -0.63996238  0.13103010  0.74852597 -1.10435414  0.78352354 
         13          14          15          16          17          18 
 3.22469291 -0.98623876  0.32818923 -1.14750260  0.22333743 -1.34944537 
         19          20          21          22          23          24 
 1.98995067  0.58703566 -0.43038260 -1.52017691 -0.09651101 -1.04276847 
         25          26          27          28          29          30 
-0.61255699  3.26857905  1.75959764 -0.36242210  0.10968144 -2.77705113 
         31          32 
-2.31245034 -1.86471908 
```

rstudent()

- **Package:** stats
- **Input:**

  `model` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> rstudent(model = modello)

          1           2           3           4           5           6 
-0.58339795 -1.95178717  1.05607073  1.52661113  1.07368887 -1.87037216 
          7           8           9          10          11          12 
 0.34667588 -0.63922752  0.13107905  0.75111918 -1.10219023  0.78568685 
         13          14          15          16          17          18 
 3.27847151 -0.98303536  0.32838016 -1.14375042  0.22345192 -1.34249887 
```

```
         19          20          21          22          23          24
 2.01164323  0.58782968 -0.42991912 -1.49773238 -0.09649454 -1.03936493
         25          26          27          28          29          30
-0.61175065  3.31837107  1.76616018 -0.36212559  0.10971516 -2.76165762
         31          32
-2.27414465 -1.85104246
```

rstudent.glm()

- **Package:** stats
- **Input:**

  `model` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Description:** residui studentizzati
- **Formula:**

$$rstudent_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> rstudent.glm(model = modello)

          1           2           3           4           5           6
-0.58339795 -1.95178717  1.05607073  1.52661113  1.07368887 -1.87037216
          7           8           9          10          11          12
 0.34667588 -0.63922752  0.13107905  0.75111918 -1.10219023  0.78568685
         13          14          15          16          17          18
 3.27847151 -0.98303536  0.32838016 -1.14375042  0.22345192 -1.34249887
         19          20          21          22          23          24
 2.01164323  0.58782968 -0.42991912 -1.49773238 -0.09649454 -1.03936493
         25          26          27          28          29          30
-0.61175065  3.31837107  1.76616018 -0.36212559  0.10971516 -2.76165762
         31          32
-2.27414465 -1.85104246
```

residuals.default()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

- **Description:** residui di lavoro
- **Formula:**

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> residuals.default(object = modello)
```

```
          1           2           3           4           5           6
-0.20165148 -0.56413249  0.29042202  0.70199431  0.34247005 -0.43487568
          7           8           9          10          11          12
 0.16386402 -0.20455645  0.04999536  0.33172955 -0.33831611  0.32602805
         13          14          15          16          17          18
 0.87408986 -0.35912141  0.10943462 -0.40119990  0.08161077 -0.33034568
         19          20          21          22          23          24
 0.50898714  0.21924503 -0.15404144 -0.68653798 -0.03098119 -0.37430000
         25          26          27          28          29          30
-0.17573412  1.66878447  0.56630428 -0.10405228  0.04163966 -0.71290188
         31          32
-0.46243717 -0.65221412
```

residuals()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo
- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> residuals(object = modello, type = "deviance")
```

```
         1          2          3          4          5          6          7
-0.5731569 -1.9263607  1.0084275  1.4656879  1.0504241 -1.7835363  0.3309445
         8          9         10         11         12         13         14
-0.6294980  0.1280339  0.7234253 -1.0862504  0.7623113  3.0093299 -0.9610107
        15         16         17         18         19         20         21
 0.3228171 -1.1213526  0.2190303 -1.2890517  1.8466732  0.5756799 -0.4215129
        22         23         24         25         26         27         28
-1.4353411 -0.0949116 -1.0171558 -0.5990789  3.1586571  1.7215083 -0.3539304
        29         30         31         32
 0.1072073 -2.7223502 -2.0764597 -1.8101537
```

- **Example 2:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> residuals(object = modello, type = "pearson")

          1           2           3           4           5           6
-0.55281621 -1.70896773  1.05411532  1.61426859  1.10594698 -1.63620653
          7           8           9          10          11          12
 0.33963895 -0.60681668  0.12908774  0.76054544 -1.01876268  0.80079916
         13          14          15          16          17          18
 3.37862422 -0.89718790  0.32855181 -1.03693106  0.22195094 -1.21105688
         19          20          21          22          23          24
 1.98713767  0.59566971 -0.41024061 -1.22623047 -0.09441767 -0.94638261
         25          26          27          28          29          30
-0.58068913  3.82214815  1.86567606 -0.34762443  0.10794374 -2.30449201
         31          32
-1.89216663 -1.56404492
```

- **Example 3:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> residuals(object = modello, type = "working")

          1           2           3           4           5           6
-0.20165148 -0.56413249  0.29042202  0.70199431  0.34247005 -0.43487568
          7           8           9          10          11          12
 0.16386402 -0.20455645  0.04999536  0.33172955 -0.33831611  0.32602805
         13          14          15          16          17          18
 0.87408986 -0.35912141  0.10943462 -0.40119990  0.08161077 -0.33034568
         19          20          21          22          23          24
 0.50898714  0.21924503 -0.15404144 -0.68653798 -0.03098119 -0.37430000
         25          26          27          28          29          30
-0.17573412  1.66878447  0.56630428 -0.10405228  0.04163966 -0.71290188
         31          32
-0.46243717 -0.65221412
```

- **Example 4:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> residuals(object = modello, type = "response")

         1          2          3          4          5          6          7
-1.5155146 -5.1771007  3.8260153  3.7120857  3.5714619 -6.1561773  0.7039655
         8          9         10         11         12         13         14
-1.8001216  0.3333039  1.7436775 -3.0677741  1.9669451 13.0594144 -2.2414318
        15         16         17         18         19         20         21
 0.9863999 -2.6800256  0.6036240 -4.4397699  7.7579880  1.6183829 -1.0925460
        22         23         24         25         26         27         28
-2.1901791 -0.2877454 -2.3928401 -1.9188070  8.7541661  6.1464257 -1.1613656
        29         30         31         32
 0.2798258 -7.4493890 -7.7422291 -3.7506647
```

residuals.glm()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo

- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+      441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+      470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+      9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> residuals.glm(object = modello, type = "deviance")

         1          2          3          4          5          6          7
-0.5731569 -1.9263607  1.0084275  1.4656879  1.0504241 -1.7835363  0.3309445
         8          9         10         11         12         13         14
-0.6294980  0.1280339  0.7234253 -1.0862504  0.7623113  3.0093299 -0.9610107
        15         16         17         18         19         20         21
 0.3228171 -1.1213526  0.2190303 -1.2890517  1.8466732  0.5756799 -0.4215129
        22         23         24         25         26         27         28
-1.4353411 -0.0949116 -1.0171558 -0.5990789  3.1586571  1.7215083 -0.3539304
        29         30         31         32
 0.1072073 -2.7223502 -2.0764597 -1.8101537
```

- **Example 2:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+      441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+      470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+      9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> residuals.glm(object = modello, type = "pearson")
```

```
          1           2           3           4           5           6
-0.55281621 -1.70896773  1.05411532  1.61426859  1.10594698 -1.63620653
          7           8           9          10          11          12
 0.33963895 -0.60681668  0.12908774  0.76054544 -1.01876268  0.80079916
         13          14          15          16          17          18
 3.37862422 -0.89718790  0.32855181 -1.03693106  0.22195094 -1.21105688
         19          20          21          22          23          24
 1.98713767  0.59566971 -0.41024061 -1.22623047 -0.09441767 -0.94638261
         25          26          27          28          29          30
-0.58068913  3.82214815  1.86567606 -0.34762443  0.10794374 -2.30449201
         31          32
-1.89216663 -1.56404492
```

- **Example 3:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> residuals.glm(object = modello, type = "working")
```

```
          1           2           3           4           5           6
-0.20165148 -0.56413249  0.29042202  0.70199431  0.34247005 -0.43487568
          7           8           9          10          11          12
 0.16386402 -0.20455645  0.04999536  0.33172955 -0.33831611  0.32602805
         13          14          15          16          17          18
 0.87408986 -0.35912141  0.10943462 -0.40119990  0.08161077 -0.33034568
         19          20          21          22          23          24
 0.50898714  0.21924503 -0.15404144 -0.68653798 -0.03098119 -0.37430000
         25          26          27          28          29          30
-0.17573412  1.66878447  0.56630428 -0.10405228  0.04163966 -0.71290188
         31          32
-0.46243717 -0.65221412
```

- **Example 4:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> residuals.glm(object = modello, type = "response")
```

```
         1          2          3          4          5          6          7
-1.5155146 -5.1771007  3.8260153  3.7120857  3.5714619 -6.1561773  0.7039655
         8          9         10         11         12         13         14
-1.8001216  0.3333039  1.7436775 -3.0677741  1.9669451 13.0594144 -2.2414318
        15         16         17         18         19         20         21
 0.9863999 -2.6800256  0.6036240 -4.4397699  7.7579880  1.6183829 -1.0925460
        22         23         24         25         26         27         28
-2.1901791 -0.2877454 -2.3928401 -1.9188070  8.7541661  6.1464257 -1.1613656
        29         30         31         32
 0.2798258 -7.4493890 -7.7422291 -3.7506647
```

resid()

- **Package:** stats
- **Input:**

`object` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità

`type = "deviance" / "pearson" / "working" / "response"` tipo di residuo

- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+      441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+      470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+      9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> resid(object = modello, type = "deviance")

         1          2          3          4          5          6          7
-0.5731569 -1.9263607  1.0084275  1.4656879  1.0504241 -1.7835363  0.3309445
         8          9         10         11         12         13         14
-0.6294980  0.1280339  0.7234253 -1.0862504  0.7623113  3.0093299 -0.9610107
        15         16         17         18         19         20         21
 0.3228171 -1.1213526  0.2190303 -1.2890517  1.8466732  0.5756799 -0.4215129
        22         23         24         25         26         27         28
-1.4353411 -0.0949116 -1.0171558 -0.5990789  3.1586571  1.7215083 -0.3539304
        29         30         31         32
 0.1072073 -2.7223502 -2.0764597 -1.8101537
```

- **Example 2:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+      441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+      470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+      9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> resid(object = modello, type = "pearson")

          1           2           3           4           5           6
-0.55281621 -1.70896773  1.05411532  1.61426859  1.10594698 -1.63620653
          7           8           9          10          11          12
 0.33963895 -0.60681668  0.12908774  0.76054544 -1.01876268  0.80079916
         13          14          15          16          17          18
 3.37862422 -0.89718790  0.32855181 -1.03693106  0.22195094 -1.21105688
         19          20          21          22          23          24
 1.98713767  0.59566971 -0.41024061 -1.22623047 -0.09441767 -0.94638261
         25          26          27          28          29          30
-0.58068913  3.82214815  1.86567606 -0.34762443  0.10794374 -2.30449201
         31          32
-1.89216663 -1.56404492
```

- **Example 3:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> resid(object = modello, type = "working")

          1           2           3           4           5           6
-0.20165148 -0.56413249  0.29042202  0.70199431  0.34247005 -0.43487568
          7           8           9          10          11          12
 0.16386402 -0.20455645  0.04999536  0.33172955 -0.33831611  0.32602805
         13          14          15          16          17          18
 0.87408986 -0.35912141  0.10943462 -0.40119990  0.08161077 -0.33034568
         19          20          21          22          23          24
 0.50898714  0.21924503 -0.15404144 -0.68653798 -0.03098119 -0.37430000
         25          26          27          28          29          30
-0.17573412  1.66878447  0.56630428 -0.10405228  0.04163966 -0.71290188
         31          32
-0.46243717 -0.65221412
```

- **Example 4:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> resid(object = modello, type = "response")

         1          2          3          4          5          6          7
-1.5155146 -5.1771007  3.8260153  3.7120857  3.5714619 -6.1561773  0.7039655
         8          9         10         11         12         13         14
-1.8001216  0.3333039  1.7436775 -3.0677741  1.9669451 13.0594144 -2.2414318
        15         16         17         18         19         20         21
 0.9863999 -2.6800256  0.6036240 -4.4397699  7.7579880  1.6183829 -1.0925460
        22         23         24         25         26         27         28
-2.1901791 -0.2877454 -2.3928401 -1.9188070  8.7541661  6.1464257 -1.1613656
        29         30         31         32
 0.2798258 -7.4493890 -7.7422291 -3.7506647
```

weighted.residuals()

- **Package:** stats
- **Input:**

  obj modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità

- **Description:** residui pesati
- **Formula:**

$$e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> weighted.residuals(obj = modello)
```

```
         1          2          3          4          5          6          7
-0.5731569 -1.9263607  1.0084275  1.4656879  1.0504241 -1.7835363  0.3309445
         8          9         10         11         12         13         14
-0.6294980  0.1280339  0.7234253 -1.0862504  0.7623113  3.0093299 -0.9610107
        15         16         17         18         19         20         21
 0.3228171 -1.1213526  0.2190303 -1.2890517  1.8466732  0.5756799 -0.4215129
        22         23         24         25         26         27         28
-1.4353411 -0.0949116 -1.0171558 -0.5990789  3.1586571  1.7215083 -0.3539304
        29         30         31         32
 0.1072073 -2.7223502 -2.0764597 -1.8101537
```

weights()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità
- **Description:** pesi iniziali
- **Formula:**

$$\underbrace{1, 1, \ldots, 1}_{n \text{ volte}}$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> weights(object = modello)

 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
27 28 29 30 31 32
 1  1  1  1  1  1
```

df.residual()

- **Package:** stats
- **Input:**

  `object` modello di regressione di *Poisson* con $k-1$ variabili esplicative ed n unità
- **Description:** gradi di libertà della devianza residua
- **Formula:**

$$n - k$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> df.residual(object = modello)

[1] 30
```

hatvalues()

- **Package:** stats
- **Input:**

  model modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità

- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> hatvalues(model = modello)

         1          2          3          4          5          6          7
0.03731074 0.03266037 0.08132102 0.06538376 0.03883352 0.10595899 0.08459283
         8          9         10         11         12         13         14
0.03243571 0.04520986 0.06594243 0.03251736 0.05341286 0.12911084 0.05050580
        15         16         17         18         19         20         21
0.03247008 0.04505800 0.03819908 0.08750591 0.13881691 0.03831420 0.04079290
        22         23         24         25         26         27         28
0.10849868 0.03286992 0.04852097 0.04352190 0.06612878 0.04282468 0.04631162
        29         30         31         32
0.04460584 0.03900696 0.19368977 0.05766771
```

cooks.distance()

- **Package:** stats
- **Input:**

  model modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> cooks.distance(model = modello)

           1            2            3            4            5            6
0.0061516720 0.0509683838 0.0535329887 0.0975269911 0.0257068065 0.1774472070
           7            8            9           10           11           12
0.0058225056 0.0063789436 0.0004131972 0.0218593896 0.0180278945 0.0191135734
          13           14           15           16           17           18
0.9715982423 0.0225472435 0.0018721138 0.0265636449 0.0010171067 0.0770683993
          19           20           21           22           23           24
0.3695534723 0.0073497811 0.0037308438 0.1026348110 0.0001566410 0.0240012884
```

```
          25           26           27           28           29           30
0.0080207542 0.5538620110 0.0813492551 0.0030765755 0.0002847026 0.1121558914
          31           32
0.5333239875 0.0794315456
```

cookd()

- **Package:** `car`
- **Input:**

  `model` modello di regressione di *Poisson* con $k - 1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(551, 651, 832, 375, 715, 868, 271, 630, 491, 372, 645,
+     441, 895, 458, 642, 492, 543, 842, 905, 542, 522, 122, 657,
+     470, 738, 371, 735, 749, 495, 716, 952, 417)
> y <- c(6, 4, 17, 9, 14, 8, 5, 7, 7, 7, 6, 8, 28, 4, 10, 4, 8,
+     9, 23, 9, 6, 1, 9, 4, 9, 14, 17, 10, 7, 3, 9, 2)
> modello <- glm(formula = y ~ x, family = poisson(link = "log"))
> cookd(model = modello)

           1            2            3            4            5            6
0.0061516720 0.0509683838 0.0535329887 0.0975269911 0.0257068065 0.1774472070
           7            8            9           10           11           12
0.0058225056 0.0063789436 0.0004131972 0.0218593896 0.0180278945 0.0191135734
          13           14           15           16           17           18
0.9715982423 0.0225472435 0.0018721138 0.0265636449 0.0010171067 0.0770683993
          19           20           21           22           23           24
0.3695534723 0.0073497811 0.0037308438 0.1026348110 0.0001566410 0.0240012884
          25           26           27           28           29           30
0.0080207542 0.5538620110 0.0813492551 0.0030765755 0.0002847026 0.1121558914
          31           32
0.5333239875 0.0794315456
```

Capitolo 22

Regressione Gamma

22.1 Simbologia

$$1\,/\,\mu_i = \beta_1 + \beta_2\, x_{i1} + \beta_3\, x_{i2} + \cdots + \beta_k\, x_{ik-1} \qquad Y_i \sim \text{Gamma}(\omega, \omega\,/\,\mu_i) \quad \forall\, i\,=\,1,\,2,\,\ldots,\,n$$

- valori osservati: $y_i \quad \forall\, i\,=\,1,\,2,\,\ldots,\,n$
- matrice del modello di dimensione $n \times k$: $\quad X$
- numero di parametri da stimare e rango della matrice del modello: $\quad k$
- numero di unità: $\quad n$
- i-esima riga della matrice del modello : $\quad X_i\,=\,(1,\,x_{i1},\,x_{i2},\ldots,\,x_{ik-1}) \quad \forall\, i\,=\,1,\,2,\,\ldots,\,n$
- vettore numerico positivo dei pesi IWLS: $\quad w\,=\,(w_1,\,w_2,\,\ldots,\,w_n)$
- matrice diagonale dei pesi IWLS di dimensione $n \times n$: $\quad W\,=\,\text{diag}(w_1^{-1},\,w_2^{-1},\,\ldots,\,w_n^{-1})$
- matrice di proiezione di dimensione $n \times n$: $\quad H\,=\,X\,(X^T\,W^{-1}\,X)^{-1}\,X^T\,W^{-1}$
- valori di leva: $\quad h_i\,=\,H_{i,i} \quad \forall\, i\,=\,1,\,2,\,\ldots,\,n$
- distanza di *Cook*: $\quad cd_i\,=\,\left(e_i^P\right)^2\,\frac{h_i}{\hat{\phi}^2\,k\,(1-h_i)^2} \quad \forall\, i\,=\,1,\,2,\,\ldots,\,n$
- stime IWLS: $\quad \hat{\beta}\,=\,\left(\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_k\right)^T$
- standard error delle stime IWLS: $\quad s_{\hat{\beta}}\,=\,\hat{\phi}\,\sqrt{\text{diag}((X^T\,W^{-1}\,X)^{-1})}$
- z-values delle stime IWLS: $\quad z_{\hat{\beta}}\,=\,\hat{\beta}\,/\,s_{\hat{\beta}}$
- correlazione delle stime IWLS: $\quad r_{\hat{\beta}_i\,\hat{\beta}_j}\,=\,\frac{\hat{\phi}^2\,(X^T\,W^{-1}\,X)^{-1}_{i,\,j}}{s_{\hat{\beta}_i}\,s_{\hat{\beta}_j}} \quad \forall\, i, j\,=\,1,\,2,\,\ldots,\,k$
- stima del parametro di dispersione: $\quad \hat{\phi}^2\,=\,\frac{1}{n-k}\,\sum_{i=1}^n\,\left(e_i^P\right)^2\,=\,\frac{1}{n-k}\,\sum_{i=1}^n\,(y_i - \hat{\mu}_i)^2\,/\,\hat{\mu}_i^2$
- residui di devianza: $\quad e_i\,=\,\text{sign}\,(y_i - \hat{\mu}_i)\,\sqrt{2\,\left((y_i - \hat{\mu}_i)\,/\,\hat{\mu}_i - \log\,(y_i\,/\,\hat{\mu}_i)\right)} \quad \forall\, i\,=\,1,\,2,\,\ldots,\,n$
- residui standard: $\quad rstandard_i\,=\,\frac{e_i}{\hat{\phi}\,\sqrt{1-h_i}} \quad \forall\, i\,=\,1,\,2,\,\ldots,\,n$
- residui di *Pearson*: $\quad e_i^P\,=\,(y_i - \hat{\mu}_i)\,/\,\hat{\mu}_i \quad \forall i\,=\,1,\,2,\,\ldots,\,n$
- residui di lavoro: $\quad e_i^W\,=\,-(y_i - \hat{\mu}_i)\,/\,\hat{\mu}_i^2 \quad \forall i\,=\,1,\,2,\,\ldots,\,n$
- residui di riposta: $\quad e_i^R\,=\,y_i - \hat{\mu}_i \quad \forall i\,=\,1,\,2,\,\ldots,\,n$
- log-verosimiglianza gamma: $\quad \hat{\ell}\,=\,\sum_{i=1}^n\,[\hat{\omega}\,(-y_i\,/\,\hat{\mu}_i - \log\,(\hat{\mu}_i)) + (\hat{\omega} - 1)\,\log\,(y_i) + \hat{\omega}\,\log\,(\hat{\omega}) - \log\,(\Gamma\,(\hat{\omega}))]$
- stima del parametro ω della distribuzione Gamma: $\quad \hat{\omega}\,=\,n\,/\,D$
- valori adattati: $\quad \hat{\mu}_i\,=\,\left(X_i\,\hat{\beta}\right)^{-1} \quad \forall\, i\,=\,1,\,2,\,\ldots,\,n$
- log-verosimiglianza gamma modello saturo:
  $\hat{\ell}_{saturo}\,=\,\sum_{i=1}^n\,[\hat{\omega}\,(-1 - \log\,(y_i)) + (\hat{\omega} - 1)\,\log\,(y_i) + \hat{\omega}\,\log\,(\hat{\omega}) - \log\,(\Gamma\,(\hat{\omega}))]$

- devianza residua: $D = 2\,\hat{\omega}^{-1}\left(\hat{\ell}_{saturo} - \hat{\ell}\right) = 2\sum_{i=1}^{n}\left[(y_i - \hat{\mu}_i)\,/\,\hat{\mu}_i - \log\left(y_i\,/\,\hat{\mu}_i\right)\right] = \sum_{i=1}^{n} e_i^2$
- gradi di libertà della devianza residua: $n - k$
- log-verosimiglianza gamma modello nullo:
  $\hat{\ell}_{nullo} = \sum_{i=1}^{n}\left[\hat{\omega}\,(-y_i\,/\,\bar{y} - \log(\bar{y})) + (\hat{\omega} - 1)\,\log(y_i) + \hat{\omega}\,\log(\hat{\omega}) - \log(\Gamma(\hat{\omega}))\right]$
- valori adattati modello nullo: $\hat{\mu} = \bar{y} \quad \forall\, i = 1, 2, \ldots, n$
- devianza residua modello nullo: $D_{nullo} = 2\,\hat{\omega}^{-1}\left(\hat{\ell}_{saturo} - \hat{\ell}_{nullo}\right)$
- gradi di libertà della devianza residua modello nullo: $n - 1$
- stima IWLS intercetta modello nullo: $\hat{\beta}_{nullo} = 1\,/\,\bar{y}$

22.2 Stima

glm()

- **Package:** `stats`
- **Input:**

  `formula` modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

  `family = Gamma(link="inverse")` famiglia e link del modello

  `x = TRUE` matrice del modello

- **Description:** analisi di regressione gamma
- **Output:**

  `coefficients` stime IWLS

  `residuals` residui di lavoro

  `fitted.values` valori adattati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` 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` valori osservati

  `x` matrice del modello

- **Formula:**

  `coefficients`

  $$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

  `residuals`

  $$e_i^W \quad \forall\, i = 1, 2, \ldots, n$$

  `fitted.values`

  $$\hat{\mu}_i \quad \forall\, i = 1, 2, \ldots, n$$

  `rank`

  $$k$$

  `linear.predictors`

  $$X\,\hat{\beta}$$

deviance

$$D$$

aic

$$-2\,\hat{\ell} + 2\,(k+1)$$

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

$$X$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"),
+     x = TRUE)
> modello$coefficients

(Intercept)           x
-0.01655439  0.01534312

> modello$residuals

            1             2             3             4             5
 3.219110e-04 -1.669382e-03 -1.245097e-03 -8.626330e-04  1.353051e-03
            6             7             8             9
-4.456480e-05  1.314954e-03  1.879616e-03  1.414317e-03

> modello$fitted.values

        1         2         3         4         5         6         7         8
122.85903  53.26389  40.00713  34.00264  28.06578  24.97221  21.61432  19.73182
        9
 18.48317

> modello$rank

[1] 2

> modello$linear.predictors

         1          2          3          4          5          6          7
0.00813941 0.01877444 0.02499554 0.02940948 0.03563058 0.04004452 0.04626563
         8          9
0.05067957 0.05410327
```

```
> modello$deviance

[1] 0.01672967

> modello$aic

[1] 37.9899

> modello$null.deviance

[1] 3.512826

> modello$weights

         1          2          3          4          5          6          7
15094.6872  2837.0712  1600.5833  1156.1874   787.6926   623.6144   467.1808
         8          9
  389.3463   341.6289

> modello$prior.weights

1 2 3 4 5 6 7 8 9
1 1 1 1 1 1 1 1 1

> modello$df.residual

[1] 7

> modello$df.null

[1] 8

> modello$y

  1   2   3   4   5   6   7   8   9
118  58  42  35  27  25  21  19  18

> modello$x

  (Intercept)        x
1           1 1.609438
2           1 2.302585
3           1 2.708050
4           1 2.995732
5           1 3.401197
6           1 3.688879
7           1 4.094345
8           1 4.382027
9           1 4.605170
attr(,"assign")
[1] 0 1
```

summary.glm()

- **Package:** stats
- **Input:**

  object modello di regressione gamma con $k-1$ variabili esplicative ed n unità

  correlation = TRUE correlazione delle stime IWLS
- **Description:** analisi di regressione gamma
- **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 IWLS non scalata

  cov.scaled matrice di covarianza delle stime IWLS scalata

  correlation matrice di correlazione delle stime IWLS
- **Formula:**

  deviance
  $$D$$
  aic
  $$-2\,\hat{\ell} + 2\,(k+1)$$
  df.residual
  $$n-k$$
  null.deviance
  $$D_{nullo}$$
  df.null
  $$n-1$$
  deviance.resid
  $$e_j \quad \forall\, j \,=\, 1,\, 2,\, \dots,\, k$$
  coefficients
  $$\hat{\beta}_j \quad s_{\hat{\beta}_j} \quad z_{\hat{\beta}_j} \quad p\text{-value} \,=\, 2\,\Phi(-\,|\,z_{\hat{\beta}_j}\,|) \qquad \forall\, j \,=\, 1,\, 2,\, \dots,\, k$$
  cov.unscaled
  $$(X^T\, W^{-1}\, X)^{-1}$$
  cov.scaled
  $$\hat{\phi}^2\,(X^T\, W^{-1}\, X)^{-1}$$
  correlation
  $$r_{\hat{\beta}_i\, \hat{\beta}_j} \quad \forall\, i, j \,=\, 1,\, 2,\, \dots,\, k$$
- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+      4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> res <- summary.glm(object = modello, correlation = TRUE)
> res$deviance

[1] 0.01672967
```

```
> res$aic

[1] 37.9899

> res$df.residual

[1] 7

> res$null.deviance

[1] 3.512826

> res$df.null

[1] 8

> res$deviance.resid

           1            2            3            4            5            6
-0.040083434  0.086411120  0.049008874  0.029049825 -0.038466050  0.001112469
           7            8            9
-0.028695647 -0.037556945 -0.026372375

> res$coefficients

               Estimate   Std. Error   t value     Pr(>|t|)
(Intercept) -0.01655439 0.0009275454 -17.84752 4.279105e-07
x            0.01534312 0.0004149591  36.97501 2.751164e-09

> res$cov.unscaled

              (Intercept)             x
(Intercept)  0.0003517261 -0.0001474395
x           -0.0001474395  0.0000703955

> res$cov.scaled

              (Intercept)             x
(Intercept)  8.603405e-07 -3.606447e-07
x           -3.606447e-07  1.721911e-07

> res$correlation

            (Intercept)         x
(Intercept)    1.000000 -0.936999
x             -0.936999  1.000000
```

glm.fit()

- **Package:** stats
- **Input:**

  `x` matrice del modello

  `y` valori osservati

  `family = Gamma(link="inverse")` famiglia e link del modello

- **Description:** analisi di regressione gamma
- **Output:**

  `coefficients` stime IWLS

  `residuals` residui di lavoro

  `fitted.values` valori adattati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` 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` valori osservati

- **Formula:**

  `coefficients`

  $$\hat{\beta}_j \quad \forall\, j = 1, 2, \dots, k$$

  `residuals`

  $$e_i^W \quad \forall\, i = 1, 2, \dots, n$$

  `fitted.values`

  $$\hat{\mu}_i \quad \forall\, i = 1, 2, \dots, n$$

  `rank`

  $$k$$

  `linear.predictors`

  $$X\,\hat{\beta}$$

  `deviance`

  $$D$$

  `aic`

  $$-2\,\hat{\ell} + 2\,(k+1)$$

  `null.deviance`

  $$D_{nullo}$$

  `weights`

  $$w_i \quad \forall\, i = 1, 2, \dots, n$$

  `prior.weights`

  $$\underbrace{1, 1, \dots, 1}_{n \text{ volte}}$$

  `df.residual`

  $$n - k$$

  `df.null`

  $$n - 1$$

y

$$y_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> X <- model.matrix(object = modello)
> res <- glm.fit(x = X, y, family = Gamma(link = "inverse"))
> res$coefficients

(Intercept)           x
-0.01655439  0.01534312

> res$residuals

[1]  3.219110e-04 -1.669382e-03 -1.245097e-03 -8.626330e-04  1.353051e-03
[6] -4.456480e-05  1.314954e-03  1.879616e-03  1.414317e-03

> res$fitted.values

[1] 122.85903  53.26389  40.00713  34.00264  28.06578  24.97221  21.61432
[8]  19.73182  18.48317

> res$rank

[1] 2

> res$linear.predictors

[1] 0.00813941 0.01877444 0.02499554 0.02940948 0.03563058 0.04004452 0.04626563
[8] 0.05067957 0.05410327

> res$deviance

[1] 0.01672967

> res$aic

[1] 37.9899

> res$null.deviance

[1] 3.512826

> res$weights

[1] 15094.6872  2837.0712  1600.5833  1156.1874   787.6926   623.6144   467.1808
[8]   389.3463   341.6289

> res$prior.weights

[1] 1 1 1 1 1 1 1 1 1

> res$df.residual

[1] 7
```

```
> res$df.null

[1] 8

> res$y

[1] 118  58  42  35  27  25  21  19  18
```

vcov()

- **Package:** stats
- **Input:**

  object modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

- **Description:** matrice di covarianza delle stime IWLS
- **Formula:**

$$\hat{\phi}^2 \, (X^T \, W^{-1} \, X)^{-1}$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> vcov(object = modello)

              (Intercept)             x
(Intercept)  8.603405e-07 -3.606447e-07
x           -3.606447e-07  1.721911e-07
```

coef()

- **Package:** stats
- **Input:**

  object modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall \, j \, = \, 1, \, 2, \, \ldots, \, k$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> coef(object = modello)

(Intercept)           x
-0.01655439  0.01534312
```

coefficients()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> coefficients(object = modello)
```

```
(Intercept)           x
-0.01655439  0.01534312
```

predict.glm()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` standard error delle stime

- **Description:** previsione
- **Output:**

  `fit` valore previsto

  `se.fit` standard error delle stime

  `residual.scale` radice quadrata della stima del parametro di dispersione

- **Formula:**

  `fit`

$$x_0^T\,\hat{\beta}$$

  `se.fit`

$$\hat{\phi}\,\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

  `residual.scale`

$$\hat{\phi}$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> predict.glm(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
```

```
$fit
          1
0.003391666

$se.fit
[1] 0.0004622413

$residual.scale
[1] 0.04945758

> res <- predict.glm(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit

          1
0.003391666

> res$se.fit

[1] 0.0004622413

> res$residual.scale

[1] 0.04945758
```

predict()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k-1$ variabili esplicative ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` standard error delle stime

- **Description:** previsione
- **Output:**

  `fit` valore previsto

  `se.fit` standard error delle stime

  `residual.scale` radice quadrata della stima del parametro di dispersione

- **Formula:**

  `fit`

$$x_0^T\,\hat{\beta}$$

  `se.fit`

$$\hat{\phi}\,\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

  `residual.scale`

$$\hat{\phi}$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> predict(object = modello, newdata = data.frame(x = 1.3), se.fit = TRUE)
```

```
$fit
          1
0.003391666

$se.fit
[1] 0.0004622413

$residual.scale
[1] 0.04945758
```

```
> res <- predict(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit
```

```
          1
0.003391666
```

```
> res$se.fit
```

```
[1] 0.0004622413
```

```
> res$residual.scale
```

```
[1] 0.04945758
```

fitted()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{\mu}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> fitted(object = modello)
```

```
        1         2         3         4         5         6         7         8
122.85903  53.26389  40.00713  34.00264  28.06578  24.97221  21.61432  19.73182
        9
 18.48317
```

fitted.values()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k-1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{\mu}_i \quad \forall\, i \,=\, 1,\, 2,\, \ldots,\, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> fitted.values(object = modello)

        1         2         3         4         5         6         7         8
122.85903  53.26389  40.00713  34.00264  28.06578  24.97221  21.61432  19.73182
        9
 18.48317
```

cov2cor()

- **Package:** stats
- **Input:**

  `V` matrice di covarianza delle stime IWLS di dimensione $k \times k$

- **Description:** converte la matrice di covarianza nella matrice di correlazione
- **Formula:**

$$r_{\hat{\beta}_i\, \hat{\beta}_j} \quad \forall\, i, j \,=\, 1,\, 2,\, \ldots,\, k$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> V <- vcov(object = modello)
> cov2cor(V)

            (Intercept)         x
(Intercept)    1.000000 -0.936999
x             -0.936999  1.000000
```

22.3 Adattamento

logLik()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k-1$ variabili esplicative ed n unità

- **Description:** log-verosimiglianza gamma

- **Formula:**

$$\hat{\ell}$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> logLik(object = modello)

'log Lik.' -15.99495 (df=3)
```

AIC()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

- **Description:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell} + 2\,(k + 1)$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> AIC(object = modello)

[1] 37.9899
```

durbin.watson()

- **Package:** car
- **Input:**

  `model` modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

- **Description:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

  `dw` valore empirico della statistica *D–W*

- **Formula:**

  `dw`

$$\sum_{i=2}^{n} (e_i - e_{i-1})^2 \,/\, D$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> durbin.watson(model = modello)
```

```
 lag Autocorrelation D-W Statistic p-value
   1       0.1835659      1.495257       0
 Alternative hypothesis: rho != 0
```

```
> res <- durbin.watson(model = modello)
> res$dw
```

```
[1] 1.495257
```

extractAIC()

- **Package:** stats
- **Input:**

  `fit` modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

- **Description:** numero di parametri del modello ed indice *AIC* generalizzato
- **Formula:**

$$k \qquad -2\,\hat{\ell} + 2\,(k+1)$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> extractAIC(fit = modello)
```

```
[1]  2.0000 37.9899
```

deviance()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

- **Description:** devianza residua
- **Formula:**

$$D$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> deviance(object = modello)
```

```
[1] 0.01672967
```

anova()

- **Package:** stats
- **Input:**

  `nullo` modello nullo di regressione gamma con n unità

  `modello` modello di regressione gamma con $k-1$ variabili esplicative con n unità

  `test = "Chisq"`
- **Description:** 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`
  $$df = k-1$$
  `Deviance`
  $$c = D_{nullo} - D$$
  `P(>|Chi|)`
  $$P(\chi^2_{df} \geq c)$$
- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> nullo <- glm(formula = y ~ 1, family = Gamma(link = "inverse"))
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> anova(nullo, modello, test = "Chisq")

Analysis of Deviance Table

Model 1: y ~ 1
Model 2: y ~ x
  Resid. Df Resid. Dev Df Deviance  P(>|Chi|)
1         8     3.5128
2         7     0.0167  1   3.4961 9.112e-313

> res <- anova(nullo, modello, test = "Chisq")
> res$"Resid. Df"

[1] 8 7

> res$"Resid. Dev"

[1] 3.51282626 0.01672967

> res$Df
```

```
[1] NA  1

> res$Deviance

[1]       NA 3.496097

> res$"P(>|Chi|)"

[1]            NA 9.111682e-313
```

22.4 Diagnostica

rstandard()

- **Package:** stats
- **Input:**

  model modello di regressione gamma con $k-1$ variabili esplicative ed n unità
- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> rstandard(model = modello)

          1           2           3           4           5           6
-2.53583145  1.87362788  1.05104455  0.62462720 -0.83312470  0.02423229
          7           8           9
-0.62991215 -0.82861703 -0.58398516
```

rstandard.glm()

- **Package:** stats
- **Input:**

  model modello di regressione gamma con $k-1$ variabili esplicative ed n unità
- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> rstandard.glm(model = modello)

          1           2           3           4           5           6
-2.53583145  1.87362788  1.05104455  0.62462720 -0.83312470  0.02423229
          7           8           9
-0.62991215 -0.82861703 -0.58398516
```

residuals.default()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k-1$ variabili esplicative ed n unità

- **Description:** residui di lavoro
- **Formula:**

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> residuals.default(object = modello)

            1             2             3             4             5
 3.219110e-04 -1.669382e-03 -1.245097e-03 -8.626330e-04  1.353051e-03
            6             7             8             9
-4.456480e-05  1.314954e-03  1.879616e-03  1.414317e-03
```

residuals()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo

- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> residuals(object = modello, type = "deviance")
```

```
          1           2           3           4           5           6
-0.040083434  0.086411120  0.049008874  0.029049825 -0.038466050  0.001112469
          7           8           9
-0.028695647 -0.037556945 -0.026372375
```

- **Example 2:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> residuals(object = modello, type = "pearson")

          1           2           3           4           5           6
-0.039549672  0.088917798  0.049812745  0.029331801 -0.037974427  0.001112881
          7           8           9
-0.028421825 -0.037088249 -0.026141052
```

- **Example 3:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> residuals(object = modello, type = "working")

            1             2             3             4             5
 3.219110e-04 -1.669382e-03 -1.245097e-03 -8.626330e-04  1.353051e-03
            6             7             8             9
-4.456480e-05  1.314954e-03  1.879616e-03  1.414317e-03
```

- **Example 4:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> residuals(object = modello, type = "response")

          1           2           3           4           5           6
-4.85903456  4.73610798  1.99286522  0.99735870 -1.06578198  0.02779111
          7           8           9
-0.61431838 -0.73181861 -0.48316949
```

residuals.glm()

- **Package:** `stats`
- **Input:**

  `object` modello di regressione gamma con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo

- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \dots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> residuals.glm(object = modello, type = "deviance")

           1            2            3            4            5            6
-0.040083434  0.086411120  0.049008874  0.029049825 -0.038466050  0.001112469
           7            8            9
-0.028695647 -0.037556945 -0.026372375
```

- **Example 2:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> residuals.glm(object = modello, type = "pearson")

           1            2            3            4            5            6
-0.039549672  0.088917798  0.049812745  0.029331801 -0.037974427  0.001112881
           7            8            9
-0.028421825 -0.037088249 -0.026141052
```

- **Example 3:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> residuals.glm(object = modello, type = "working")

            1             2             3             4             5
 3.219110e-04 -1.669382e-03 -1.245097e-03 -8.626330e-04  1.353051e-03
            6             7             8             9
-4.456480e-05  1.314954e-03  1.879616e-03  1.414317e-03
```

- **Example 4:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> residuals.glm(object = modello, type = "response")

          1           2           3           4           5           6
-4.85903456  4.73610798  1.99286522  0.99735870 -1.06578198  0.02779111
          7           8           9
-0.61431838 -0.73181861 -0.48316949
```

resid()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo
- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> resid(object = modello, type = "deviance")

           1            2            3            4            5            6
-0.040083434  0.086411120  0.049008874  0.029049825 -0.038466050  0.001112469
           7            8            9
-0.028695647 -0.037556945 -0.026372375
```

- **Example 2:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> resid(object = modello, type = "pearson")

           1            2            3            4            5            6
-0.039549672  0.088917798  0.049812745  0.029331801 -0.037974427  0.001112881
           7            8            9
-0.028421825 -0.037088249 -0.026141052
```

- **Example 3:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> resid(object = modello, type = "working")
```

```
            1             2             3             4             5
 3.219110e-04 -1.669382e-03 -1.245097e-03 -8.626330e-04  1.353051e-03
            6             7             8             9
-4.456480e-05  1.314954e-03  1.879616e-03  1.414317e-03
```

- **Example 4:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> resid(object = modello, type = "response")
```

```
          1           2           3           4           5           6
-4.85903456  4.73610798  1.99286522  0.99735870 -1.06578198  0.02779111
          7           8           9
-0.61431838 -0.73181861 -0.48316949
```

weighted.residuals()

- **Package:** stats
- **Input:**

  obj modello di regressione gamma con $k-1$ variabili esplicative ed n unità

- **Description:** residui pesati
- **Formula:**

$$e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> weighted.residuals(obj = modello)
```

```
           1            2            3            4            5            6
-0.040083434  0.086411120  0.049008874  0.029049825 -0.038466050  0.001112469
           7            8            9
-0.028695647 -0.037556945 -0.026372375
```

weights()

- **Package:** stats
- **Input:**

  object modello di regressione di gamma con $k-1$ variabili esplicative ed n unità

- **Description:** pesi iniziali
- **Formula:**

$$\underbrace{1, 1, \ldots, 1}_{n \text{ volte}}$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+      4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> weights(object = modello)

1 2 3 4 5 6 7 8 9
1 1 1 1 1 1 1 1 1
```

df.residual()

- **Package:** stats
- **Input:**

  `object` modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

- **Description:** gradi di libertà della devianza residua
- **Formula:**

$$n - k$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+      4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> df.residual(object = modello)

[1] 7
```

hatvalues()

- **Package:** stats
- **Input:**

  `model` modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+      4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> hatvalues(model = modello)

        1         2         3         4         5         6         7         8 
0.8978535 0.1304254 0.1111234 0.1157409 0.1284959 0.1383694 0.1515889 0.1601396 
        9 
0.1662629 
```

cooks.distance()

- **Package:** stats
- **Input:**

  model modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> cooks.distance(model = modello)

           1            2            3            4            5            6
2.751369e+01 2.787598e-01 7.133585e-02 2.603212e-02 4.986974e-02 4.718454e-05
           7            8            9
3.477467e-02 6.383541e-02 3.341085e-02
```

cookd()

- **Package:** car
- **Input:**

  model modello di regressione gamma con $k - 1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = Gamma(link = "inverse"))
> cookd(model = modello)

           1            2            3            4            5            6
2.751369e+01 2.787598e-01 7.133585e-02 2.603212e-02 4.986974e-02 4.718454e-05
           7            8            9
3.477467e-02 6.383541e-02 3.341085e-02
```

Capitolo 23

Regressione di Wald

23.1 Simbologia

$$1 / \mu_i^2 = \beta_1 + \beta_2\, x_{i1} + \beta_3\, x_{i2} + \cdots + \beta_k\, x_{ik-1} \qquad Y_i \sim \text{Wald}(\mu_i, \omega) \quad \forall\, i = 1, 2, \ldots, n$$

- valori osservati: $y_i \quad \forall\, i = 1, 2, \ldots, n$
- 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 : $X_i = (1, x_{i1}, x_{i2}, \ldots, x_{ik-1}) \quad \forall\, i = 1, 2, \ldots, n$
- vettore numerico positivo dei pesi IWLS: $w = (w_1, w_2, \ldots, w_n)$
- 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 $n \times n$: $H = X\,(X^T\, W^{-1}\, X)^{-1}\, X^T\, W^{-1}$
- valori di leva: $h_i = H_{i,i} \quad \forall\, i = 1, 2, \ldots, n$
- distanza di *Cook*: $cd_i = \left(e_i^P\right)^2 \frac{h_i}{\hat{\phi}^2\, k\, (1-h_i)^2} \quad \forall\, i = 1, 2, \ldots, n$
- stime IWLS: $\hat{\beta} = \left(\hat{\beta}_1, \hat{\beta}_2, \ldots, \hat{\beta}_k\right)^T$
- standard error delle stime IWLS: $s_{\hat{\beta}} = \hat{\phi}\,\sqrt{\text{diag}((X^T\, W^{-1}\, X)^{-1})}$
- z-values delle stime IWLS: $z_{\hat{\beta}} = \hat{\beta} \,/\, s_{\hat{\beta}}$
- correlazione delle stime IWLS: $r_{\hat{\beta}_i\, \hat{\beta}_j} = \frac{\hat{\phi}^2\, (X^T\, W^{-1}\, X)^{-1}_{i,\,j}}{s_{\hat{\beta}_i}\, s_{\hat{\beta}_j}} \quad \forall\, i, j = 1, 2, \ldots, k$
- stima del parametro di dispersione: $\hat{\phi}^2 = \frac{1}{n-k} \sum_{i=1}^{n} \left(e_i^P\right)^2 = \frac{1}{n-k} \sum_{i=1}^{n} \left((y_i - \hat{\mu}_i) \,/\, \hat{\mu}_i^{3\,/\,2}\right)^2$
- residui di devianza: $e_i = \text{sign}\,(y_i - \hat{\mu}_i)\, \sqrt{(y_i - \hat{\mu}_i)^2 \,/\, (y_i\, \hat{\mu}_i^2)} \quad \forall\, i = 1, 2, \ldots, n$
- residui standard: $rstandard_i = \frac{e_i}{\hat{\phi}\,\sqrt{1-h_i}} \quad \forall\, i = 1, 2, \ldots, n$
- residui di *Pearson*: $e_i^P = (y_i - \hat{\mu}_i) \,/\, \hat{\mu}_i^{3\,/\,2} \quad \forall i = 1, 2, \ldots, n$
- residui di lavoro: $e_i^W = -2\, (y_i - \hat{\mu}_i) \,/\, \hat{\mu}_i^3 \quad \forall i = 1, 2, \ldots, n$
- residui di riposta: $e_i^R = y_i - \hat{\mu}_i \quad \forall i = 1, 2, \ldots, n$
- log-verosimiglianza normale inversa: $\hat{\ell} = \frac{n}{2} \log\,(\hat{\omega}) - \frac{3}{2} \sum_{i=1}^{n} \log\,(2\,\pi\, y_i) - \hat{\omega} \sum_{i=1}^{n} (y_i - \hat{\mu}_i)^2 \,/\, (2\, y_i\, \hat{\mu}_i^2)$
- stima del parametro ω della distribuzione Wald: $\hat{\omega} = n \,/\, D$
- valori adattati: $\hat{\mu}_i = \left(X_i\, \hat{\beta}\right)^{-1\,/\,2} \quad \forall\, i = 1, 2, \ldots, n$
- log-verosimiglianza normale inversa modello saturo: $\hat{\ell}_{saturo} = \frac{n}{2} \log\,(\hat{\omega}) - \frac{3}{2} \sum_{i=1}^{n} \log\,(2\,\pi\, y_i)$

- devianza residua: $D = 2\,\hat{\omega}^{-1}\left(\hat{\ell}_{saturo} - \hat{\ell}\right) = \sum_{i=1}^{n}(y_i - \hat{\mu}_i)^2 / (y_i\,\hat{\mu}_i^2) = \sum_{i=1}^{n} e_i^2$
- gradi di libertà della devianza residua: $n - k$
- log-verosimiglianza normale inversa modello nullo:
  $\hat{\ell}_{nullo} = \frac{n}{2}\log(\hat{\omega}) - \frac{3}{2}\sum_{i=1}^{n}\log(2\,\pi\,y_i) - \hat{\omega}\sum_{i=1}^{n}(y_i - \bar{y})^2 / (2\,y_i\,\bar{y}^2)$
- valori adattati modello nullo: $\hat{\mu} = \bar{y} \quad \forall\, i = 1, 2, \ldots, n$
- devianza residua modello nullo: $D_{nullo} = 2\,\hat{\omega}^{-1}\left(\hat{\ell}_{saturo} - \hat{\ell}_{nullo}\right)$
- gradi di libertà della devianza residua modello nullo: $n - 1$
- stima IWLS intercetta modello nullo: $\hat{\beta}_{nullo} = 1 / \bar{y}^2$

23.2 Stima

glm()

- **Package:** `stats`
- **Input:**

  `formula` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

  `family = inverse.gaussian(link="1/mu^2")` famiglia e link del modello

  `x = TRUE` matrice del modello
- **Description:** analisi di regressione normale inversa
- **Output:**

  `coefficients` stime IWLS

  `residuals` residui di lavoro

  `fitted.values` valori adattati

  `rank` rango della matrice del modello

  `linear.predictors` predittori lineari

  `deviance` 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` valori osservati

  `x` matrice del modello
- **Formula:**

  `coefficients`
  $$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

  `residuals`
  $$e_i^W \quad \forall\, i = 1, 2, \ldots, n$$

  `fitted.values`
  $$\hat{\mu}_i \quad \forall\, i = 1, 2, \ldots, n$$

  `rank`
  $$k$$

  `linear.predictors`
  $$X\,\hat{\beta}$$

deviance

$$D$$

aic

$$-2\,\hat{\ell} + 2\,(k+1)$$

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

$$X$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"),
+     x = TRUE)
> modello$coefficients

 (Intercept)            x
-0.001107977  0.000721914

> modello$residuals

            1             2             3             4             5
 1.441199e-05 -4.052050e-04 -3.766423e-04 -2.882582e-04  2.402256e-05
            6             7             8             9
 4.397338e-05  3.595650e-04  5.697415e-04  6.762886e-04

> modello$fitted.values

        1         2         3         4         5         6         7         8
136.21078  42.47477  34.36037  30.79207  27.24286  25.35854  23.26344  22.05690
        9
 21.24028

> modello$rank

[1] 2

> modello$linear.predictors

           1            2            3            4            5            6
5.389855e-05 5.542911e-04 8.470019e-04 1.054684e-03 1.347394e-03 1.555076e-03
           7            8            9
1.847788e-03 2.055469e-03 2.216559e-03
```

```
> modello$deviance

[1] 0.006931123

> modello$aic

[1] 61.57485

> modello$null.deviance

[1] 0.08779963

> modello$weights

         1          2          3          4          5          6          7
632025.412  19157.982  10142.024   7299.044   5054.816   4076.798   3147.514
         8          9
  2682.741   2395.664

> modello$prior.weights

1 2 3 4 5 6 7 8 9
1 1 1 1 1 1 1 1 1

> modello$df.residual

[1] 7

> modello$df.null

[1] 8

> modello$y

  1   2   3   4   5   6   7   8   9
118  58  42  35  27  25  21  19  18

> modello$x

  (Intercept)        x
1           1 1.609438
2           1 2.302585
3           1 2.708050
4           1 2.995732
5           1 3.401197
6           1 3.688879
7           1 4.094345
8           1 4.382027
9           1 4.605170
attr(,"assign")
[1] 0 1
```

summary.glm()

- **Package:** stats
- **Input:**

  object modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

  correlation = TRUE correlazione delle stime IWLS

- **Description:** analisi di regressione normale inversa
- **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 IWLS non scalata

  cov.scaled matrice di covarianza delle stime IWLS scalata

  correlation matrice di correlazione delle stime IWLS

- **Formula:**

  deviance
  $$D$$

  aic
  $$-2\,\hat{\ell} + 2\,(k+1)$$

  df.residual
  $$n - k$$

  null.deviance
  $$D_{nullo}$$

  df.null
  $$n - 1$$

  deviance.resid
  $$e_j \quad \forall\, j = 1, 2, \ldots, k$$

  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
  $$\hat{\phi}^2\,(X^T\,W^{-1}\,X)^{-1}$$

  correlation
  $$r_{\hat{\beta}_i\,\hat{\beta}_j} \quad \forall\, i,j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> res <- summary.glm(object = modello, correlation = TRUE)
> res$deviance

[1] 0.006931123
```

```
> res$aic

[1] 61.57485

> res$df.residual

[1] 7

> res$null.deviance

[1] 0.08779963

> res$df.null

[1] 8

> res$deviance.resid

           1            2            3            4            5            6
-0.012307674  0.047994662  0.034307576  0.023099121 -0.001715587 -0.002827732
           7            8            9
-0.021231743 -0.031795091 -0.035957248

> res$coefficients

                Estimate   Std. Error   t value     Pr(>|t|)
(Intercept) -0.001107977 1.675366e-04 -6.613343 0.0003005580
x            0.000721914 9.468635e-05  7.624267 0.0001237599

> res$cov.unscaled

              (Intercept)             x
(Intercept)  2.549583e-05 -1.399142e-05
x           -1.399142e-05  8.143748e-06

> res$cov.scaled

              (Intercept)             x
(Intercept)  2.806852e-08 -1.540325e-08
x           -1.540325e-08  8.965505e-09

> res$correlation

            (Intercept)         x
(Intercept)    1.000000 -0.970991
x             -0.970991  1.000000
```

glm.fit()

- **Package:** stats
- **Input:**

  x matrice del modello

  y valori osservati

  family = inverse.gaussian(link="1/mu^2") famiglia e link del modello
- **Description:** analisi di regressione normale inversa
- **Output:**

  coefficients stime IWLS

  residuals residui di lavoro

  fitted.values valori adattati

  rank rango della matrice del modello

  linear.predictors predittori lineari

  deviance 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 valori osservati
- **Formula:**

  coefficients

  $$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

  residuals

  $$e_i^W \quad \forall\, i = 1, 2, \ldots, n$$

  fitted.values

  $$\hat{\mu}_i \quad \forall\, i = 1, 2, \ldots, n$$

  rank

  $$k$$

  linear.predictors

  $$X\,\hat{\beta}$$

  deviance

  $$D$$

  aic

  $$-2\,\hat{\ell} + 2\,(k+1)$$

  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

$$y_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> X <- model.matrix(object = modello)
> res <- glm.fit(x = X, y, family = Gamma(link = "inverse"))
> res$coefficients

(Intercept)           x
-0.01655439  0.01534312

> res$residuals

[1]  3.219110e-04 -1.669382e-03 -1.245097e-03 -8.626330e-04  1.353051e-03
[6] -4.456480e-05  1.314954e-03  1.879616e-03  1.414317e-03

> res$fitted.values

[1] 122.85903  53.26389  40.00713  34.00264  28.06578  24.97221  21.61432
[8]  19.73182  18.48317

> res$rank

[1] 2

> res$linear.predictors

[1] 0.00813941 0.01877444 0.02499554 0.02940948 0.03563058 0.04004452 0.04626563
[8] 0.05067957 0.05410327

> res$deviance

[1] 0.01672967

> res$aic

[1] 37.9899

> res$null.deviance

[1] 3.512826

> res$weights

[1] 15094.6872  2837.0712  1600.5833  1156.1874   787.6926   623.6144   467.1808
[8]   389.3463   341.6289

> res$prior.weights

[1] 1 1 1 1 1 1 1 1 1

> res$df.residual

[1] 7
```

```
> res$df.null
```

```
[1] 8
```

```
> res$y
```

```
[1] 118  58  42  35  27  25  21  19  18
```

vcov()

- **Package:** stats
- **Input:**

  object modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

- **Description:** matrice di covarianza delle stime IWLS
- **Formula:**

$$\hat{\phi}^2\,(X^T\,W^{-1}\,X)^{-1}$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> vcov(object = modello)
```

```
              (Intercept)             x
(Intercept)  2.806852e-08 -1.540325e-08
x           -1.540325e-08  8.965505e-09
```

coef()

- **Package:** stats
- **Input:**

  object modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> coef(object = modello)
```

```
 (Intercept)            x
-0.001107977  0.000721914
```

coefficients()

- **Package:** stats
- **Input:**

  object modello di regressione normale inversa con $k - 1$ variabili esplicative ed n unità

- **Description:** stime IWLS
- **Formula:**

$$\hat{\beta}_j \quad \forall\, j = 1, 2, \dots, k$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+      4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> coefficients(object = modello)

 (Intercept)             x
-0.001107977  0.000721914
```

predict.glm()

- **Package:** stats
- **Input:**

  object modello di regressione normale inversa con $k - 1$ variabili esplicative ed n unità

  newdata il valore di x_0

  se.fit = TRUE standard error delle stime

- **Description:** previsione
- **Output:**

  fit valore previsto

  se.fit standard error delle stime

  residual.scale radice quadrata della stima del parametro di dispersione

- **Formula:**

  fit

$$x_0^T \, \hat{\beta}$$

  se.fit

$$\hat{\phi} \, \sqrt{x_0^T \, (X^T \, W^{-1} \, X)^{-1} \, x_0}$$

  residual.scale

$$\hat{\phi}$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+      4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> predict.glm(object = modello, newdata = data.frame(x = 1.3),
+      se.fit = TRUE)
```

```
$fit
            1
-0.0001694891

$se.fit
[1] 5.631855e-05

$residual.scale
[1] 0.03317991

> res <- predict.glm(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit

            1
-0.0001694891

> res$se.fit

[1] 5.631855e-05

> res$residual.scale

[1] 0.03317991
```

predict()

- **Package:** stats
- **Input:**

  `object` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

  `newdata` il valore di x_0

  `se.fit = TRUE` standard error delle stime
- **Description:** previsione
- **Output:**

  `fit` valore previsto

  `se.fit` standard error delle stime

  `residual.scale` radice quadrata della stima del parametro di dispersione
- **Formula:**

  `fit`

  $$x_0^T\,\hat{\beta}$$

  `se.fit`

  $$\hat{\phi}\,\sqrt{x_0^T\,(X^T\,W^{-1}\,X)^{-1}\,x_0}$$

  `residual.scale`

  $$\hat{\phi}$$
- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> predict(object = modello, newdata = data.frame(x = 1.3), se.fit = TRUE)
```

```
$fit
            1
-0.0001694891

$se.fit
[1] 5.631855e-05

$residual.scale
[1] 0.03317991
```

```
> res <- predict(object = modello, newdata = data.frame(x = 1.3),
+     se.fit = TRUE)
> res$fit
```

```
            1
-0.0001694891
```

```
> res$se.fit
```

```
[1] 5.631855e-05
```

```
> res$residual.scale
```

```
[1] 0.03317991
```

fitted()

- **Package:** stats
- **Input:**

  object modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{\mu}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> fitted(object = modello)
```

```
        1         2         3         4         5         6         7         8
136.21078  42.47477  34.36037  30.79207  27.24286  25.35854  23.26344  22.05690
        9
 21.24028
```

fitted.values()

- **Package:** stats
- **Input:**

  `object` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

- **Description:** valori adattati
- **Formula:**

$$\hat{\mu}_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> fitted.values(object = modello)

        1         2         3         4         5         6         7         8
136.21078  42.47477  34.36037  30.79207  27.24286  25.35854  23.26344  22.05690
        9
 21.24028
```

cov2cor()

- **Package:** stats
- **Input:**

  `V` matrice di covarianza delle stime IWLS di dimensione $k \times k$

- **Description:** converte la matrice di covarianza nella matrice di correlazione
- **Formula:**

$$r_{\hat{\beta}_i\, \hat{\beta}_j} \quad \forall\, i, j = 1, 2, \ldots, k$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> V <- vcov(object = modello)
> cov2cor(V)

            (Intercept)         x
(Intercept)    1.000000 -0.970991
x             -0.970991  1.000000
```

23.3 Adattamento

logLik()

- **Package:** stats
- **Input:**

  `object` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

- **Description:** log-verosimiglianza normale inversa

- **Formula:**

$$\hat{\ell}$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> logLik(object = modello)

'log Lik.' -27.78742 (df=3)
```

AIC()

- **Package:** stats
- **Input:**

  object modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

- **Description:** indice *AIC*
- **Formula:**

$$-2\,\hat{\ell}+2\,(k+1)$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> AIC(object = modello)

[1] 61.57485
```

durbin.watson()

- **Package:** car
- **Input:**

  model modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

- **Description:** test di *Durbin–Watson* per verificare la presenza di autocorrelazioni tra i residui
- **Output:**

  dw valore empirico della statistica *D–W*

- **Formula:**

  dw

$$\sum_{i=2}^{n}(e_i-e_{i-1})^2\,/\,D$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> durbin.watson(model = modello)
```

```
 lag Autocorrelation D-W Statistic p-value
   1       0.5326615     0.7262834       0
 Alternative hypothesis: rho != 0
```

```
> res <- durbin.watson(model = modello)
> res$dw
```

```
[1] 0.7262834
```

extractAIC()

- **Package:** stats
- **Input:**

  `fit` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità
- **Description:** numero di parametri del modello ed indice *AIC* generalizzato
- **Formula:**

$$k \qquad -2\,\hat{\ell}+2\,(k+1)$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> extractAIC(fit = modello)
```

```
[1]  2.00000 61.57485
```

deviance()

- **Package:** stats
- **Input:**

  `object` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità
- **Description:** devianza residua
- **Formula:**

$$D$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> deviance(object = modello)
```

```
[1] 0.006931123
```

anova()

- **Package:** stats
- **Input:**

  `nullo` modello nullo di regressione normale inversa con n unità

  `modello` modello di regressione normale inversa con $k-1$ variabili esplicative con n unità

  `test = "Chisq"`
- **Description:** 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`
  $$df = k-1$$
  `Deviance`
  $$c = D_{nullo} - D$$
  `P(>|Chi|)`
  $$P(\chi^2_{df} \geq c)$$
- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> nullo <- glm(formula = y ~ 1, family = inverse.gaussian(link = "1/mu^2"))
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> anova(nullo, modello, test = "Chisq")

Analysis of Deviance Table

Model 1: y ~ 1
Model 2: y ~ x
  Resid. Df Resid. Dev Df Deviance P(>|Chi|)
1         8   0.087800
2         7   0.006931  1 0.080869 1.029e-17

> res <- anova(nullo, modello, test = "Chisq")
> res$"Resid. Df"

[1] 8 7

> res$"Resid. Dev"

[1] 0.087799631 0.006931123

> res$Df
```

```
[1] NA  1

> res$Deviance

[1]        NA 0.0808685

> res$"P(>|Chi|)"

[1]           NA 1.028899e-17
```

23.4 Diagnostica

rstandard()

- **Package:** stats
- **Input:**

  model modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità
- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> rstandard(model = modello)

          1           2           3           4           5           6
-2.77015888  1.50909106  1.08734334  0.73698543 -0.05524365 -0.09162823
          7           8           9
-0.69379244 -1.04490257 -1.18674607
```

rstandard.glm()

- **Package:** stats
- **Input:**

  model modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità
- **Description:** residui standard
- **Formula:**

$$rstandard_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> rstandard.glm(model = modello)

          1           2           3           4           5           6
-2.77015888  1.50909106  1.08734334  0.73698543 -0.05524365 -0.09162823
          7           8           9
-0.69379244 -1.04490257 -1.18674607
```

residuals.default()

- **Package:** stats
- **Input:**

  `object` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

- **Description:** residui di lavoro
- **Formula:**

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> residuals.default(object = modello)
```

```
            1             2             3             4             5
 1.441199e-05 -4.052050e-04 -3.766423e-04 -2.882582e-04  2.402256e-05
            6             7             8             9
 4.397338e-05  3.595650e-04  5.697415e-04  6.762886e-04
```

residuals()

- **Package:** stats
- **Input:**

  `object` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo

- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> residuals(object = modello, type = "deviance")
```

```
           1            2            3            4            5            6
-0.012307674  0.047994662  0.034307576  0.023099121 -0.001715587 -0.002827732
           7            8            9
-0.021231743 -0.031795091 -0.035957248
```

- **Example 2:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> residuals(object = modello, type = "pearson")

           1            2            3            4            5            6
-0.011455426  0.056084313  0.037930257  0.024626916 -0.001707923 -0.002807670
           7            8            9
-0.020172435 -0.029509689 -0.033101109
```

- **Example 3:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> residuals(object = modello, type = "working")

            1             2             3             4             5
 1.441199e-05 -4.052050e-04 -3.766423e-04 -2.882582e-04  2.402256e-05
            6             7             8             9
 4.397338e-05  3.595650e-04  5.697415e-04  6.762886e-04
```

- **Example 4:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> residuals(object = modello, type = "response")

          1           2           3           4           5           6
-18.2107760  15.5252280   7.6396327   4.2079288  -0.2428551  -0.3585357
          7           8           9
 -2.2634414  -3.0569010  -3.2402835
```

residuals.glm()

- **Package:** `stats`
- **Input:**

  `object` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo
- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> residuals.glm(object = modello, type = "deviance")

           1            2            3            4            5            6
-0.012307674  0.047994662  0.034307576  0.023099121 -0.001715587 -0.002827732
           7            8            9
-0.021231743 -0.031795091 -0.035957248
```

- **Example 2:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> residuals.glm(object = modello, type = "pearson")

           1            2            3            4            5            6
-0.011455426  0.056084313  0.037930257  0.024626916 -0.001707923 -0.002807670
           7            8            9
-0.020172435 -0.029509689 -0.033101109
```

- **Example 3:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> residuals.glm(object = modello, type = "working")

            1             2             3             4             5
 1.441199e-05 -4.052050e-04 -3.766423e-04 -2.882582e-04  2.402256e-05
            6             7             8             9
 4.397338e-05  3.595650e-04  5.697415e-04  6.762886e-04
```

- **Example 4:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> residuals.glm(object = modello, type = "response")

          1           2           3           4           5           6
-18.2107760  15.5252280   7.6396327   4.2079288  -0.2428551  -0.3585357
          7           8           9
 -2.2634414  -3.0569010  -3.2402835
```

resid()

- **Package:** stats
- **Input:**

  `object` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

  `type = "deviance" / "pearson" / "working" / "response"` tipo di residuo

- **Description:** residui
- **Formula:**

`type = "deviance"`

$$e_i \quad \forall i = 1, 2, \ldots, n$$

`type = "pearson"`

$$e_i^P \quad \forall i = 1, 2, \ldots, n$$

`type = "working"`

$$e_i^W \quad \forall i = 1, 2, \ldots, n$$

`type = "response"`

$$e_i^R \quad \forall i = 1, 2, \ldots, n$$

- **Example 1:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> resid(object = modello, type = "deviance")

           1            2            3            4            5            6
-0.012307674  0.047994662  0.034307576  0.023099121 -0.001715587 -0.002827732
           7            8            9
-0.021231743 -0.031795091 -0.035957248
```

- **Example 2:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> resid(object = modello, type = "pearson")

           1            2            3            4            5            6
-0.011455426  0.056084313  0.037930257  0.024626916 -0.001707923 -0.002807670
           7            8            9
-0.020172435 -0.029509689 -0.033101109
```

- **Example 3:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> resid(object = modello, type = "working")
```

```
           1            2            3            4            5 
1.441199e-05 -4.052050e-04 -3.766423e-04 -2.882582e-04 2.402256e-05
           6            7            8            9 
4.397338e-05  3.595650e-04  5.697415e-04  6.762886e-04
```

- **Example 4:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> resid(object = modello, type = "response")

          1           2           3           4           5           6 
-18.2107760  15.5252280   7.6396327   4.2079288  -0.2428551  -0.3585357 
          7           8           9 
 -2.2634414  -3.0569010  -3.2402835
```

weighted.residuals()

- **Package:** stats
- **Input:**

  `obj` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

- **Description:** residui pesati
- **Formula:**

$$e_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> weighted.residuals(obj = modello)

           1            2            3            4            5            6 
-0.012307674  0.047994662  0.034307576  0.023099121 -0.001715587 -0.002827732 
           7            8            9 
-0.021231743 -0.031795091 -0.035957248
```

weights()

- **Package:** stats
- **Input:**

  `object` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

- **Description:** pesi iniziali
- **Formula:**

$$\underbrace{1, 1, \ldots, 1}_{n \text{ volte}}$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> weights(object = modello)
```

```
1 2 3 4 5 6 7 8 9
1 1 1 1 1 1 1 1 1
```

df.residual()

- **Package:** stats
- **Input:**

  `object` modello di regressione normale inversa con $k - 1$ variabili esplicative ed n unità

- **Description:** gradi di libertà della devianza residua
- **Formula:**

$$n - k$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> df.residual(object = modello)
```

```
[1] 7
```

hatvalues()

- **Package:** stats
- **Input:**

  `model` modello di regressione normale inversa con $k - 1$ variabili esplicative ed n unità

- **Description:** valori di leva
- **Formula:**

$$h_i \quad \forall i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+     4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> hatvalues(model = modello)
```

```
         1          2          3          4          5          6          7
0.98206951 0.08123487 0.09573399 0.10767587 0.12398794 0.13489803 0.14932884
         8          9
0.15895722 0.16611374
```

cooks.distance()

- **Package:** stats
- **Input:**

  `model` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+      4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> cooks.distance(model = modello)

           1            2            3            4            5            6
1.820539e+02 1.374788e-01 7.650060e-02 3.724884e-02 2.140500e-04 6.453313e-04
           7            8            9
3.813787e-02 8.887771e-02 1.188766e-01
```

cookd()

- **Package:** car
- **Input:**

  `model` modello di regressione normale inversa con $k-1$ variabili esplicative ed n unità

- **Description:** distanza di *Cook*
- **Formula:**

$$cd_i \quad \forall\, i = 1, 2, \ldots, n$$

- **Examples:**

```
> x <- c(1.609438, 2.302585, 2.70805, 2.995732, 3.401197, 3.688879,
+      4.094345, 4.382027, 4.60517)
> y <- c(118, 58, 42, 35, 27, 25, 21, 19, 18)
> modello <- glm(formula = y ~ x, family = inverse.gaussian(link = "1/mu^2"))
> cookd(model = modello)

           1            2            3            4            5            6
1.820539e+02 1.374788e-01 7.650060e-02 3.724884e-02 2.140500e-04 6.453313e-04
           7            8            9
3.813787e-02 8.887771e-02 1.188766e-01
```

Parte VI

Appendice

Appendice A

Packages

Package	Descrizione	Status	Versione
`actuar`	Actuarial functions	Not Installed	0.9-7
`base`	The R Base Package	Loaded	2.7.0
`boot`	Bootstrap R (S-Plus) Functions (Canty)	Not Loaded	1.2-32
`BSDA`	Basic Statistics and Data Analysis	Not Installed	0.1
`car`	Companion to Applied Regression	Not Installed	1.2-7
`corpcor`	Efficient Estimation of Covariance and (Partial) Correlation	Not Installed	1.4.7
`datasets`	The R Datasets Package	Loaded	2.7.0
`distributions`	Probability distributions based on TI-83 Plus	Not Installed	1.4
`e1071`	Misc Functions of the Department of Statistics (e1071), TU Wien	Not Installed	1.5-17
`formularioR`	Formulario di Statistica con R	Not Installed	1.0
`faraway`	Functions and datasets for books by Julian Faraway.	Not Installed	1.0.3
`fBasics`	Rmetrics - Markets and Basic Statistics	Not Installed	240.10068.1
`foreign`	Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, dBase, ...	Not Loaded	0.8-25
`fUtilities`	Rmetrics - Rmetrics Function Utilities	Not Installed	270.73
`graphics`	The R Graphics Package	Loaded	2.7.0
`grDevices`	The R Graphics Devices and Support for Colours and Fonts	Loaded	2.7.0
`gtools`	Various R programming tools	Not Installed	2.4.0

ineq	Measuring inequality, concentration and poverty	Not Installed	0.2-8
labstatR	Libreria del Laboratorio di Statistica con R	Not Installed	1.0.4
leaps	regression subset selection	Not Installed	2.7
lmtest	Testing Linear Regression Models	Not Installed	0.9-21
MASS	Main Package of Venables and Ripley's MASS	Not Loaded	7.2-41
MCMCpack	Markov chain Monte Carlo (MCMC) Package	Not Installed	0.9-4
methods	Formal Methods and Classes	Loaded	2.7.0
moments	Moments, cumulants, skewness, kurtosis and related tests	Not Installed	0.11
MPV	Data Sets from Montgomery, Peck and Vining's Book	Not Installed	1.25
mvtnorm	Multivariate Normal and T Distribution	Not Installed	0.8-1
nlme	Linear and Nonlinear Mixed Effects Models	Not Loaded	3.1-88
nortest	Tests for Normality	Not Installed	1.0
pastecs	Package for Analysis of Space-Time Ecological Series	Not Installed	1.3-4
Rcmdr	R Commander	Not Installed	1.3-11
schoolmath	Functions and datasets for math used in school	Not Installed	0.2
sigma2tools	Test of hypothesis about sigma2	Not Installed	1.2.6
stats	The R Stats Package	Loaded	2.7.0
strucchange	Testing, Monitoring and Dating Structural Changes	Not Installed	1.3-2
SuppDists	Supplementary distributions	Not Installed	1.1-2
tseries	Time series analysis and computational finance	Not Installed	0.10-13
UsingR	Data sets for the text Using R for Introductory Statistics	Not Installed	0.1-8
utils	The R Utils Package	Loaded	2.7.0

Download Packages from CRAN site

Appendice B

Links

R site search

Site search	http://finzi.psych.upenn.edu/search.html
Mailing list archives	http://tolstoy.newcastle.edu.au/R/
Help center	http://www.stat.ucl.ac.be/ISdidactique/Rhelp/
Help for R (Jonathan Baron)	http://finzi.psych.upenn.edu/
r-help mailing list information	http://www.mail-archive.com/r-help@stat.math.ethz.ch/info.html

R information

CRAN	http://cran.r-project.org/
Web site	http://www.r-project.org/
News	http://cran.r-project.org/doc/Rnews/
R Wiki	http://wiki.r-project.org/
Bioconductor	http://www.bioconductor.org/

R GUIs

Projects (CRAN)	http://www.sciviews.org/_rgui/
R Commander	http://socserv.socsci.mcmaster.ca/jfox/Misc/Rcmdr/index.html
Rpad	http://www.rpad.org/Rpad/
SciViews	http://www.sciviews.org/SciViews-R/

JGR	http://stats.math.uni-augsburg.de/JGR/

Tinn-R

SourceForge (main)	http://sourceforge.net/projects/tinn-r
SciViews	http://www.sciviews.org/Tinn-R

Statistics

Journal of Statistical Software	http://www.jstatsoft.org/
HyperStat Text Book	http://davidmlane.com/hyperstat/index.html
Electronic Textbook StatSoft	http://www.statsoftinc.com/textbook/stathome.html

Processing

Miktex	http://miktex.org/
Deplate	http://deplate.sourceforge.net/index.php
Txt2tags	http://txt2tags.sourceforge.net/

Bibliografia

Agostinelli C. (2000). Introduzione ad R. Published on the URL: http://www.dst.unive.it/~laboratorior/doc/materiale/unaintroduzioneadR.pdf.

Bashir S. (2004). Getting Started in R. Published on the URL: http://www.sbtc.ltd.uk/notes/Rintro.pdf.

Boggiani R. (2004). Introduzione ad R. Published on the URL: http://digilander.libero.it/robicox/manuali/pdf/mainr.pdf.

Brazzale A.; Chiogna M.; Gaetan C.; Sartori N. (2001). 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/.

Crawley M. (2007). *The R book*. Wiley, England.

Crivellari F. (2006). *Analisi Statistica dei dati con R*. APOGEO, Milano.

D'Agostini G. (2005). 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.

Dalgaard P. (2002). *Introductory Statistics with R*. Springer-Verlag, New York.

Dell'Omodarme M. (2007). Alcune note su R. Published on the URL: http://www.cran.r-project.org/doc/contrib/DellOmodarme-esercitazioni-R.pdf.

Faraway J. (2002). Practical Regression and Anova using R. Published on the URL: http://www.cran.r-project.org/doc/contrib/Faraway-PRA.pdf.

Fox J. (2002). *An R and S-Plus Companion to Applied Regression*. SAGE Pubblications, Thousand Oaks, California.

Green C. (2004). The Stat 390 R Primer. Published on the URL: http://www.stat.washington.edu/cggreen/rprimer/rprimer.pdf.

Højsgaard S. (2005). 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.

Iacus S.; Masarotto G. (2007). *Laboratorio di statistica con R*. McGraw-Hill, Milano, seconda edizione.

Kim D.-Y. (2004). R Tutorial. Published on the URL: http://www.math.ilstu.edu/dhkim/Rstuff/Rtutor.html, Department of Mathematics Illinois State University.

Lemon J. (2005). Kickstarting R. Published on the URL: http://www.cran.r-project.org/doc/contrib/Lemon-kickstart/index.html.

Maindonald J. H. (2004). 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.

Mineo A. M. (2003). Una guida all'utilizzo dell'ambiente statistico R. Published on the URL: http://www.cran.r-project.org/doc/contrib/Mineo-dispensaR.pdf.

Muggeo V. M. R. (2002). Il linguaggio R: concetti introduttivi ed esempi. Published on the URL: http://www.cran.r-project.org/doc/contrib/nozioniR.pdf.

Owen W. J. (2006). The R Guide. Published on the URL: http://cran.r-project.org/doc/contrib/Owen-TheRGuide.pdf.

Paradis E. (2002). R for beginners. Published on the URL: http://cran.r-project.org/doc/contrib/Paradis-rdebuts_en.pdf.

Parpinel F. (2000). La statistica applicata attraverso l'uso del programma R. Published on the URL: http://venus.unive.it/statcomp/r/man_Parpinel.pdf.

Polettini S. (2004). Introduzione ad R. Published on the URL: http://www.dipstat.unina.it/stat_appl/labo1.pdf.

Pollice A. (2000). 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.

Ricci V. (2004). ANALISI DELLE SERIE STORICHE CON R. Published on the URL: http://www.cran.r-project.org/doc/contrib/Ricci-ts-italian.pdf.

Robinson A. (2006). Objects in R. Published on the URL: http://www.forestry.ubc.ca/biometrics/documents/R-Workshop/objects.pdf.

Scott T. (2004). An Introduction to R. Published on the URL: http://www.mc.vanderbilt.edu/gcrc/workshop_files/2004-08-20.pdf.

Scrucca L. (2005). 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.

Soliani L. (2005). Manuale di Statistica per la Ricerca e la Professione. Published on the URL: http://www.dsa.unipr.it/soliani/soliani.html.

Stefanini F. M. (2007). *INTRODUZIONE ALLA STATISTICA APPLICATA con esempi in R.* PEARSON Education, Milano.

Tancredi A. (2005). 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.

Venables W. N.; Ripley B. D. (2002). *Modern Applied Statistics with S.* Springer-Verlag, New York.

Verzani J. (2002). Using R for Introductory Statistics. Published on the URL: http://www.cran.r-project.org/doc/contrib/Verzani-SimpleR.pdf.

Indice analitico