STATISTICA E ANALISI DEI DATI

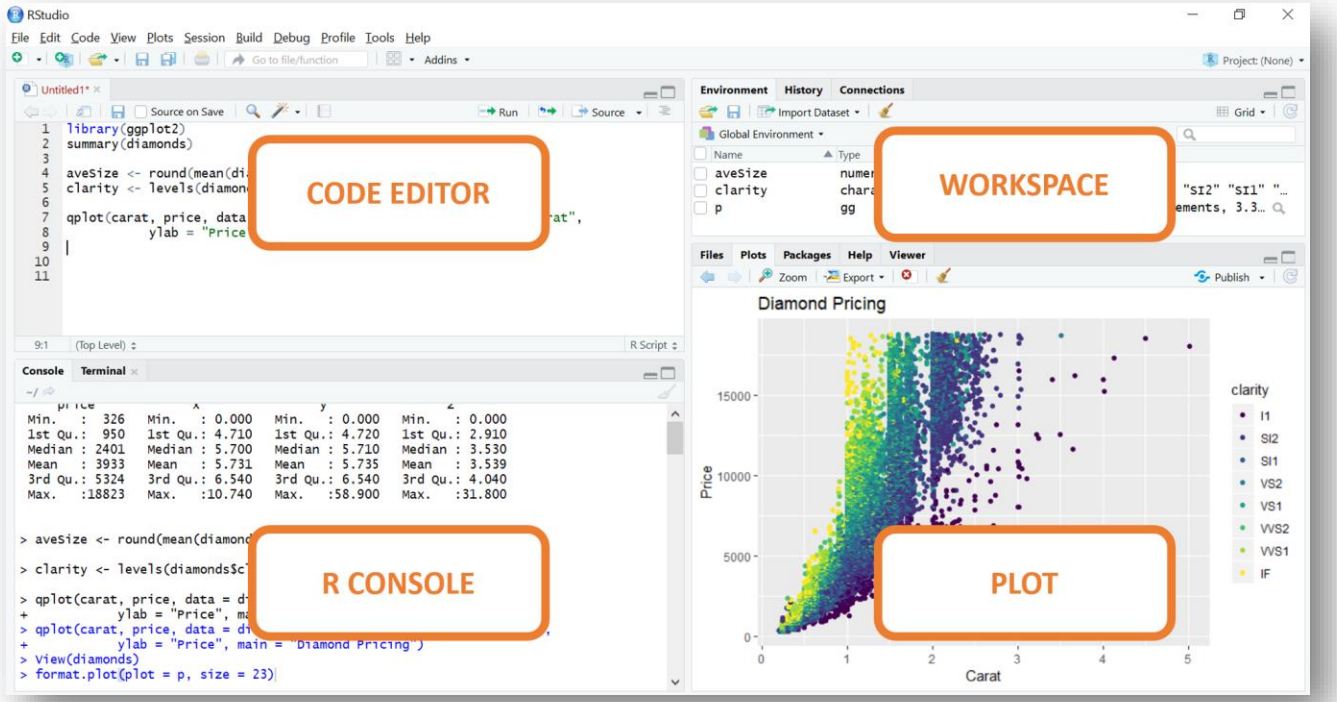
Capitolo 3 - Rappresentazioni grafiche dei dati

Dott. Stefano Cirillo
Dott. Luigi Di Biasi

a.a. 2023-2024

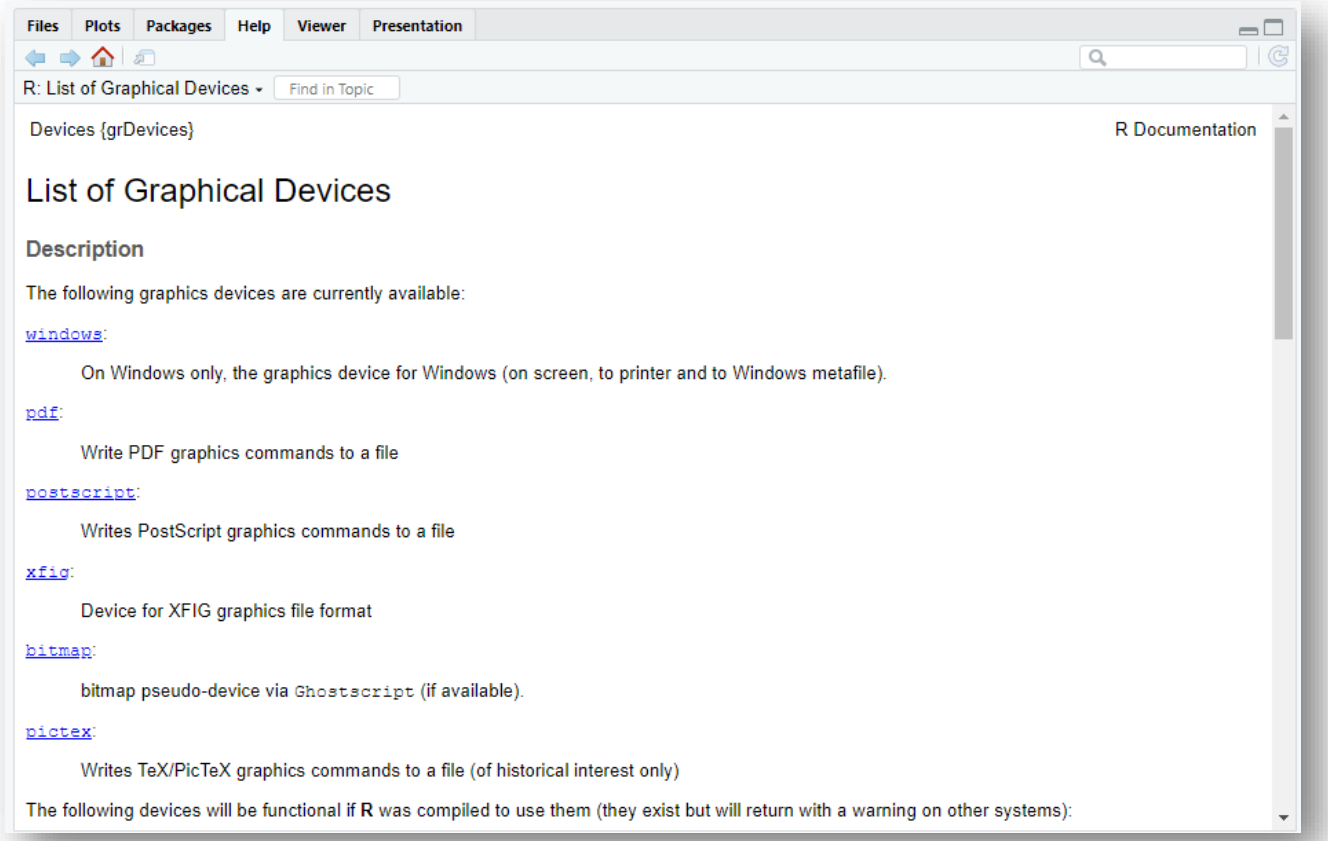
Introduzione

- In Rstudio ci sono 4 aree principali:
  - **Code Editor** (editor di testo per scrivere ed eseguire lo script);
  - **R Console** (riga di comando, esegue i comandi in modalità batch e mostra output);
  - **Workspace** (informazioni riguardo gli oggetti creati, cronologia dei comandi);
  - **Plot** (mostra gli output grafici, help)
- L'esecuzione di un **comando grafico** attiva automaticamente una **finestra grafica (*Plot*)**
  - Mostrerà l'output dei diversi comandi grafici eseguiti dalla linea di comando



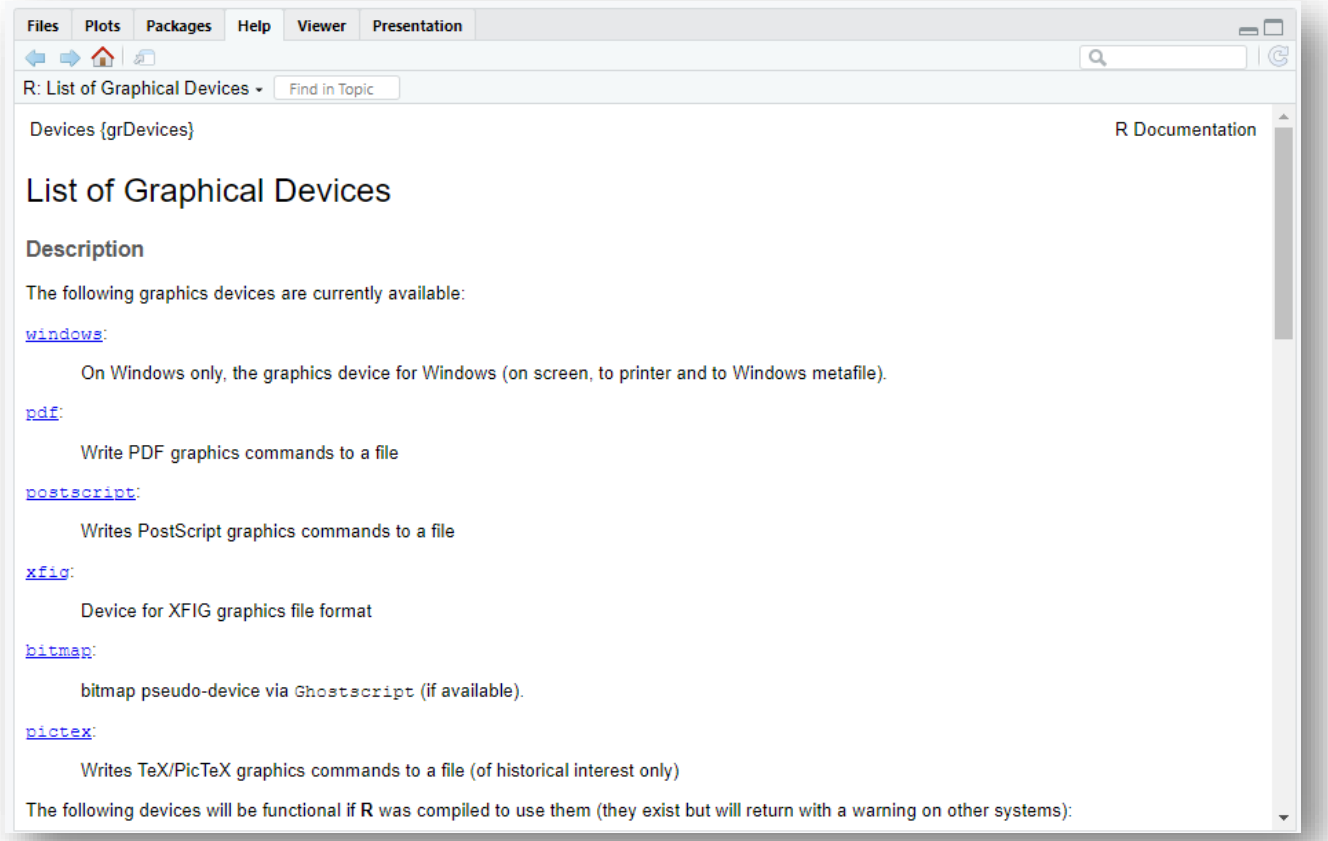
Introduzione

- Le dimensioni di ogni grafico prodotto possono essere modificate trascinando il mouse
- Ogni nuovo grafico creato cancellerà quello precedentemente realizzato
  - **Suggerimento**: Salvare sempre il grafico risultante in un file
  - Per salvare un'immagine presente nella finestra grafica è necessario utilizzare i menu di Rstudio e scegliere il formato dell'immagine da salvare



Introduzione

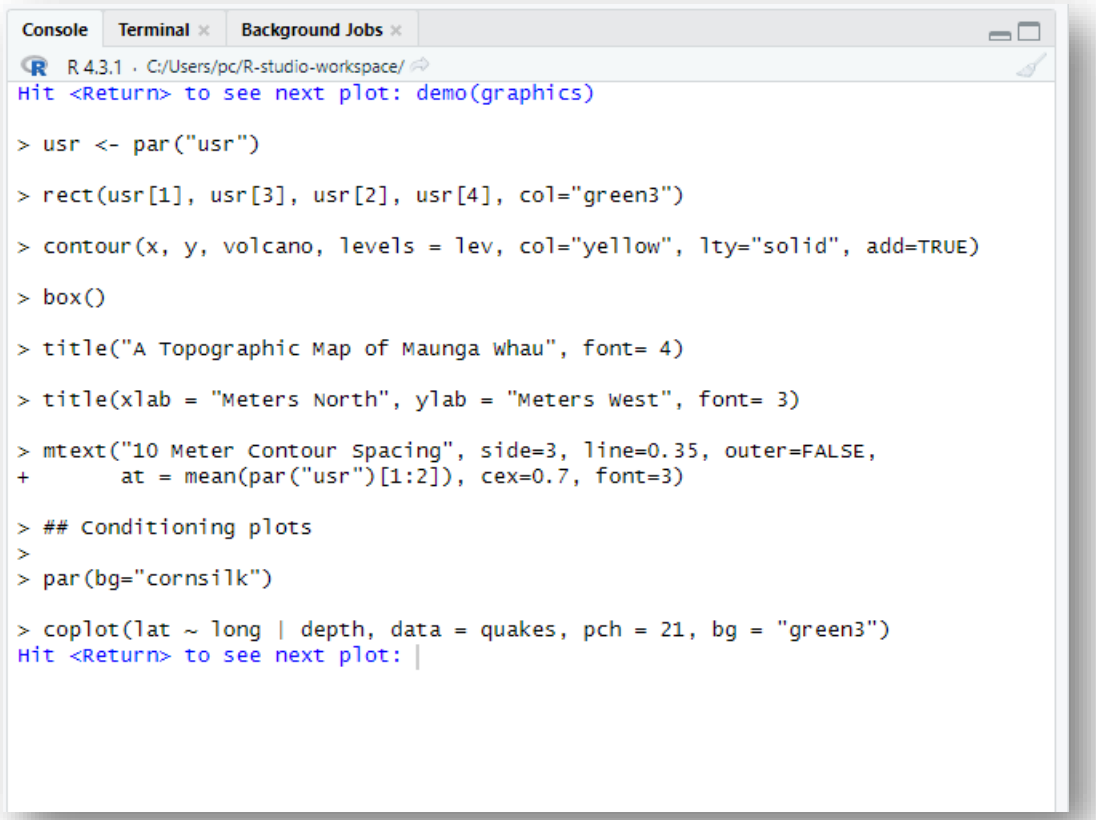
- Le dimensioni di ogni grafico prodotto possono essere modificate trascinando il mouse
- Ogni nuovo grafico creato cancellerà quello precedentemente realizzato
  - **Suggerimento**: Salvare sempre il grafico risultante in un file
  - Per salvare un'immagine presente nella finestra grafica è necessario utilizzare i menu di Rstudio e scegliere il formato dell'immagine da salvare
- Comandi Utili:
  - **?device** : visualizza la lista dei formati grafici possibili
  - **dev.off()** : I file creati restano attivi finch´e non si chiude la sessione di R



Funzioni Grafiche

- Le funzioni grafiche possono essere divise in tre gruppi:
  - **funzioni ad alto livello:** creano un nuovo grafico nella finestra grafica e agiscono in modo automatico sulla base dei soli parametri indicati negli argomenti;
  - **funzioni a basso livello:** aggiungono particolari ad un grafico già esistente;
  - **funzioni per grafici interattivi:** consentono di aggiungere o estrarre interattivamente informazioni da un grafico già esistente.
- Per visualizzare un esempio di funzioni grafiche in R si esegua in console:

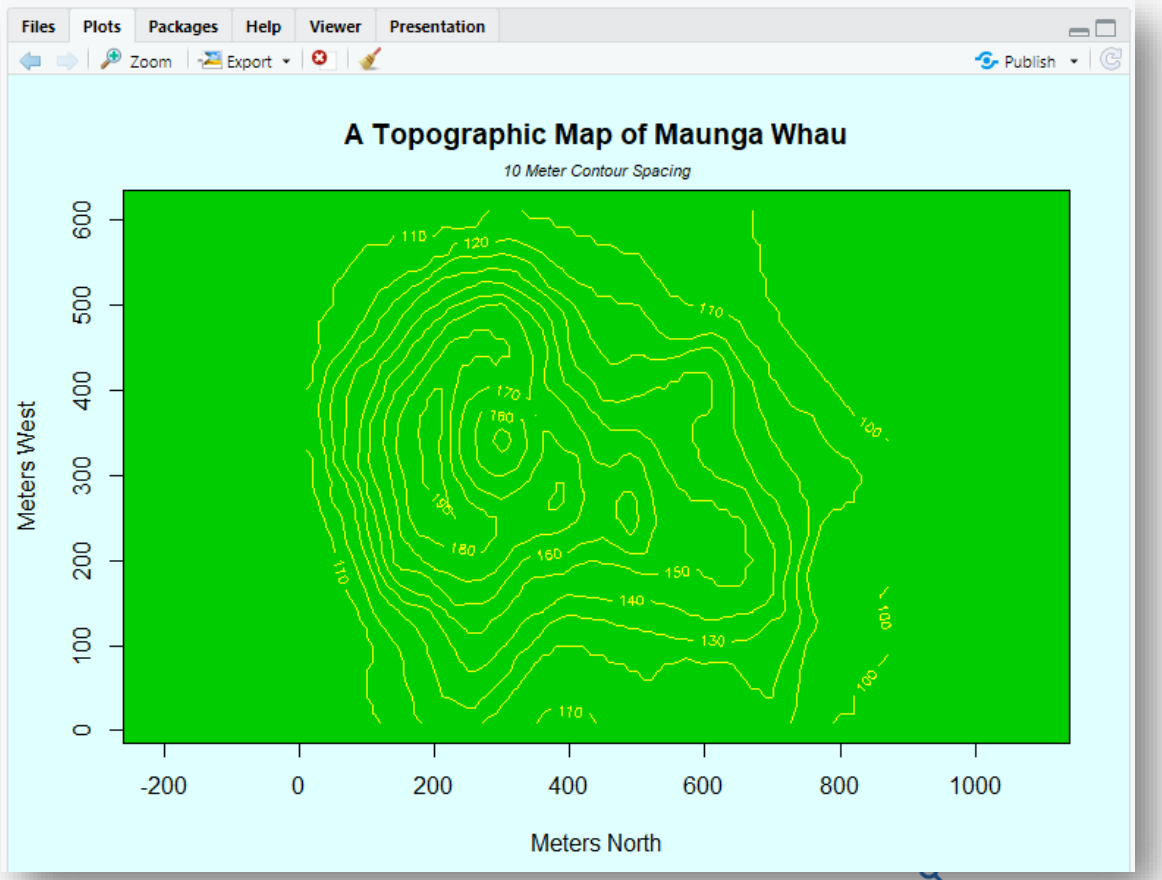
  > demo(graphics)



Funzioni Grafiche

- Le funzioni grafiche possono essere divise in tre gruppi:
  - **funzioni ad alto livello:** creano un nuovo grafico nella finestra grafica e agiscono in modo automatico sulla base dei soli parametri indicati negli argomenti;
  - **funzioni a basso livello:** aggiungono particolari ad un grafico già esistente;
  - **funzioni per grafici interattivi:** consentono di aggiungere o estrarre interattivamente informazioni da un grafico già esistente.
- Per visualizzare un esempio di funzioni grafiche in R si esegua in console:

  > demo(graphics)

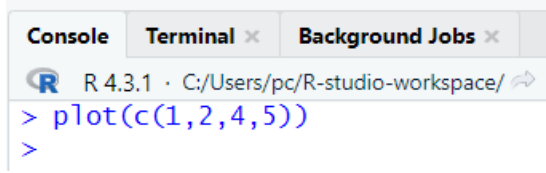


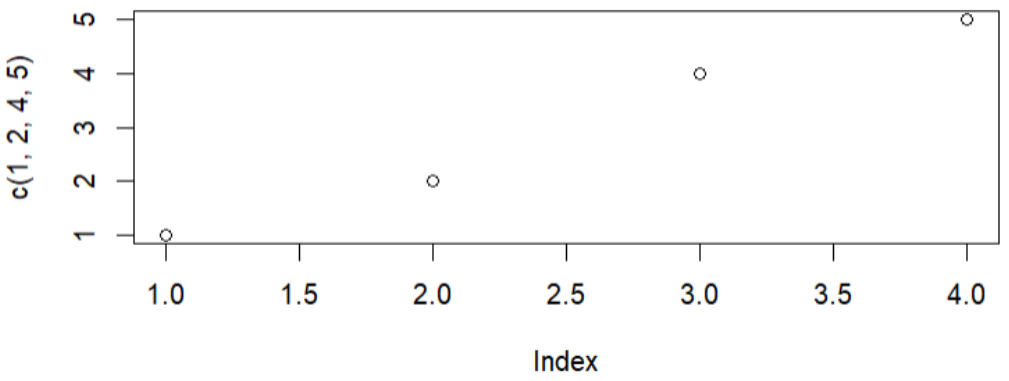
Comandi di Alto Livello

plot(x)	Se x è un vettore, il grafico illustra l'andamento dei valori assunti dal vettore rispetto ai rispettivi indici; se x è un vettore di numeri complessi, il grafico illustra l'andamento delle parti immaginarie rispetto alle parti reali.
plot(f)	Se f è di tipo fattore, viene prodotto un grafico a barre che illustra l'andamento delle frequenze dei valori contenuti nella variabile.
plot(x, y)	Se x è y sono vettori, il grafico produce uno scatterplot (nuvola di punti) di y rispetto a x.
plot(f, y)	Se f è un fattore e y un vettore numerico, la funzione produce un boxplot di y per ogni livello di f.
plot(m)	Se m è una matrice a due colonne produce uno scatterplot di una colonna rispetto all'altra.
plot(df)	Se df è un data frame, viene prodotto un grafico distribuzionale delle variabili del data frame.
plot(~expr)	Se expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico distribuzionale degli oggetti indicati.
plot(y~expr)	Se y è un oggetto e expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico di y rispetto a ciascun oggetto indicato in expr.

Comandi di Alto Livello

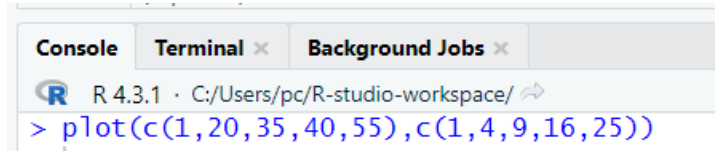
plot(x)	Se x è un vettore, il grafico illustra l'andamento dei valori assunti dal vettore rispetto ai rispettivi indici; se x è un vettore di numeri complessi, il grafico illustra l'andamento delle parti immaginarie rispetto alle parti reali.
plot(f)	Se f è di tipo fattore, viene prodotto un grafico a barre che illustra l'andamento delle frequenze dei valori contenuti nella variabile.
plot(x, y)	Se x è y sono vettori, il grafico produce uno scatterplot (nuvola di punti) di y rispetto a x.
plot(f, y)	Se f è un fattore e y un vettore numerico, la funzione produce un boxplot di y per ogni livello di f.
plot(m)	Se m è una matrice a due colonne produce uno scatterplot di una colonna rispetto all'altra.
plot(df)	Se df è un data frame, viene prodotto un grafico distribuzionale delle variabili del data frame.
plot(˜expr)	Se expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico distribuzionale degli oggetti indicati.
plot(y˜expr)	Se y è un oggetto e expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico di y rispetto a ciascun oggetto indicato in expr.

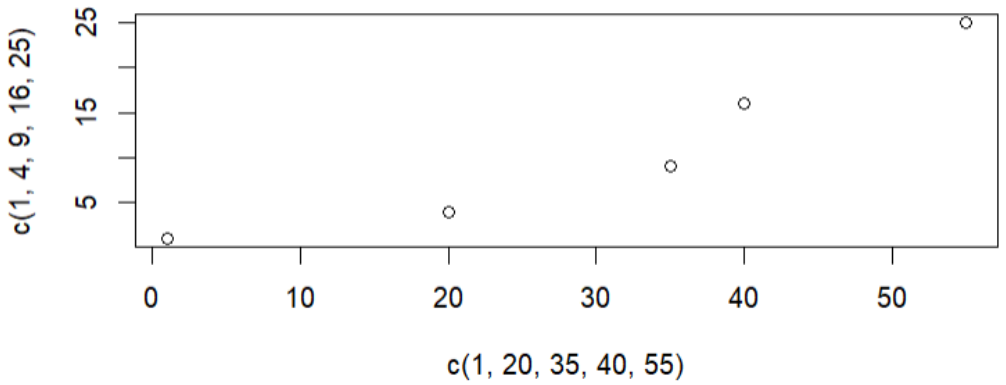




Comandi di Alto Livello

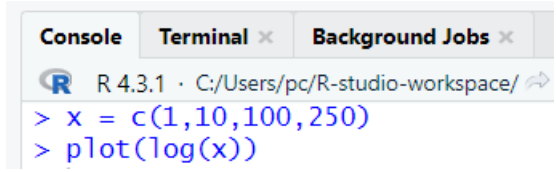
plot(x)	Se x è un vettore, il grafico illustra l'andamento dei valori assunti dal vettore rispetto ai rispettivi indici; se x è un vettore di numeri complessi, il grafico illustra l'andamento delle parti immaginarie rispetto alle parti reali.
plot(f)	Se f è di tipo fattore, viene prodotto un grafico a barre che illustra l'andamento delle frequenze dei valori contenuti nella variabile.
plot(x, y)	Se x è y sono vettori, il grafico produce uno scatterplot (nuvola di punti) di y rispetto a x.
plot(f, y)	Se f è un fattore e y un vettore numerico, la funzione produce un boxplot di y per ogni livello di f.
plot(m)	Se m è una matrice a due colonne produce uno scatterplot di una colonna rispetto all'altra.
plot(df)	Se df è un data frame, viene prodotto un grafico distribuzionale delle variabili del data frame.
plot(˜expr)	Se expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico distribuzionale degli oggetti indicati.
plot(y˜expr)	Se y è un oggetto e expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico di y rispetto a ciascun oggetto indicato in expr.

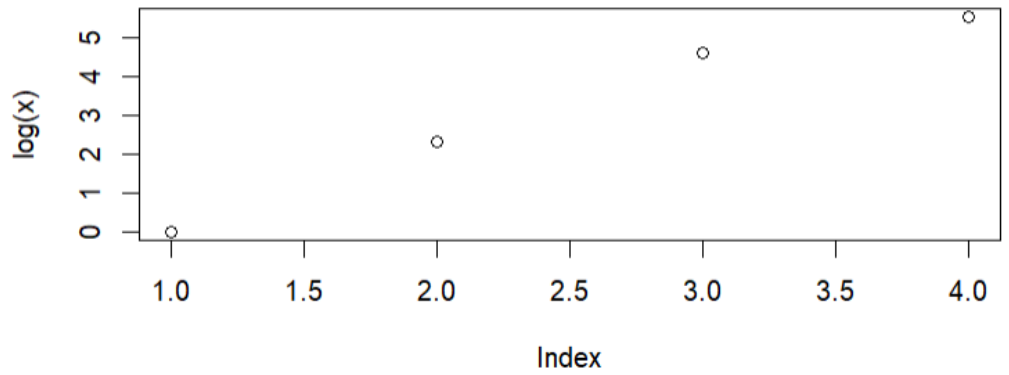




Comandi di Alto Livello

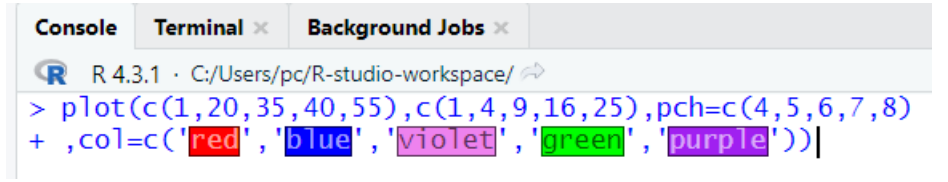
plot(x)	Se x è un vettore, il grafico illustra l'andamento dei valori assunti dal vettore rispetto ai rispettivi indici; se x è un vettore di numeri complessi, il grafico illustra l'andamento delle parti immaginarie rispetto alle parti reali.
plot(f)	Se f è di tipo fattore, viene prodotto un grafico a barre che illustra l'andamento delle frequenze dei valori contenuti nella variabile.
plot(x, y)	Se x è y sono vettori, il grafico produce uno scatterplot (nuvola di punti) di y rispetto a x.
plot(f, y)	Se f è un fattore e y un vettore numerico, la funzione produce un boxplot di y per ogni livello di f.
plot(m)	Se m è una matrice a due colonne produce uno scatterplot di una colonna rispetto all'altra.
plot(df)	Se df è un data frame, viene prodotto un grafico distribuzionale delle variabili del data frame.
plot(˜expr)	Se expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico distribuzionale degli oggetti indicati.
plot(y˜expr)	Se y è un oggetto e expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico di y rispetto a ciascun oggetto indicato in expr.

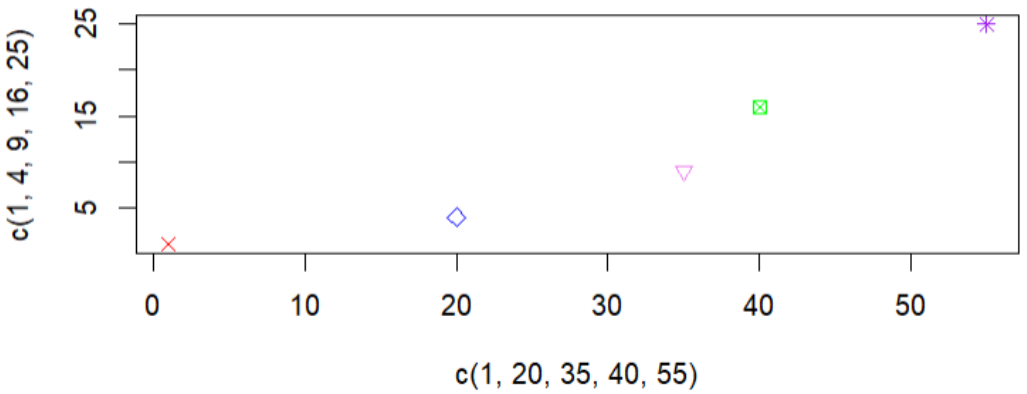




Comandi di Alto Livello

plot(x)	Se x è un vettore, il grafico illustra l'andamento dei valori assunti dal vettore rispetto ai rispettivi indici; se x è un vettore di numeri complessi, il grafico illustra l'andamento delle parti immaginarie rispetto alle parti reali.
plot(f)	Se f è di tipo fattore, viene prodotto un grafico a barre che illustra l'andamento delle frequenze dei valori contenuti nella variabile.
plot(x, y)	Se x è y sono vettori, il grafico produce uno scatterplot (nuvola di punti) di y rispetto a x.
plot(f, y)	Se f è un fattore e y un vettore numerico, la funzione produce un boxplot di y per ogni livello di f.
plot(m)	Se m è una matrice a due colonne produce uno scatterplot di una colonna rispetto all'altra.
plot(df)	Se df è un data frame, viene prodotto un grafico distribuzionale delle variabili del data frame.
plot(˜expr)	Se expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico distribuzionale degli oggetti indicati.
plot(y˜expr)	Se y è un oggetto e expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico di y rispetto a ciascun oggetto indicato in expr.

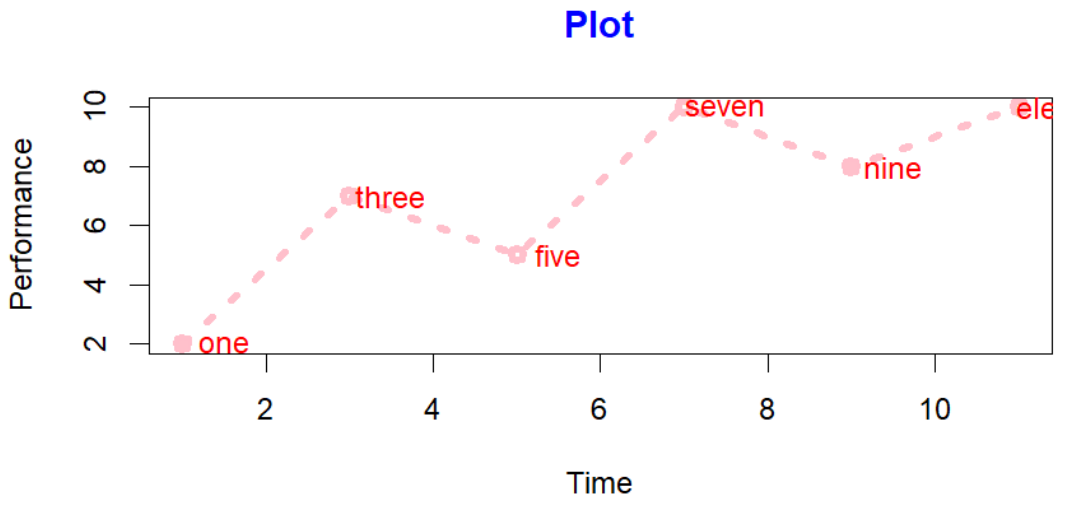




Comandi di Alto Livello

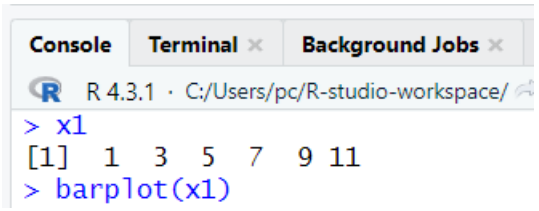
plot(x)	Se x è un vettore, il grafico illustra l'andamento dei valori assunti dal vettore rispetto ai rispettivi indici; se x è un vettore di numeri complessi, il grafico illustra l'andamento delle parti immaginarie rispetto alle parti reali.
plot(f)	Se f è di tipo fattore, viene prodotto un grafico a barre che illustra l'andamento delle frequenze dei valori contenuti nella variabile.
plot(x, y)	Se x è y sono vettori, il grafico produce uno scatterplot (nuvola di punti) di y rispetto a x.
plot(f, y)	Se f è un fattore e y un vettore numerico, la funzione produce un boxplot di y per ogni livello di f.
plot(m)	Se m è una matrice a due colonne produce uno scatterplot di una colonna rispetto all'altra.
plot(df)	Se df è un data frame, viene prodotto un grafico distribuzionale delle variabili del data frame.
plot(~expr)	Se expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico distribuzionale degli oggetti indicati.
plot(y~expr)	Se y è un oggetto e expr è una lista di nomi di oggetti separati da +, viene prodotto un grafico di y rispetto a ciascun oggetto indicato in expr.

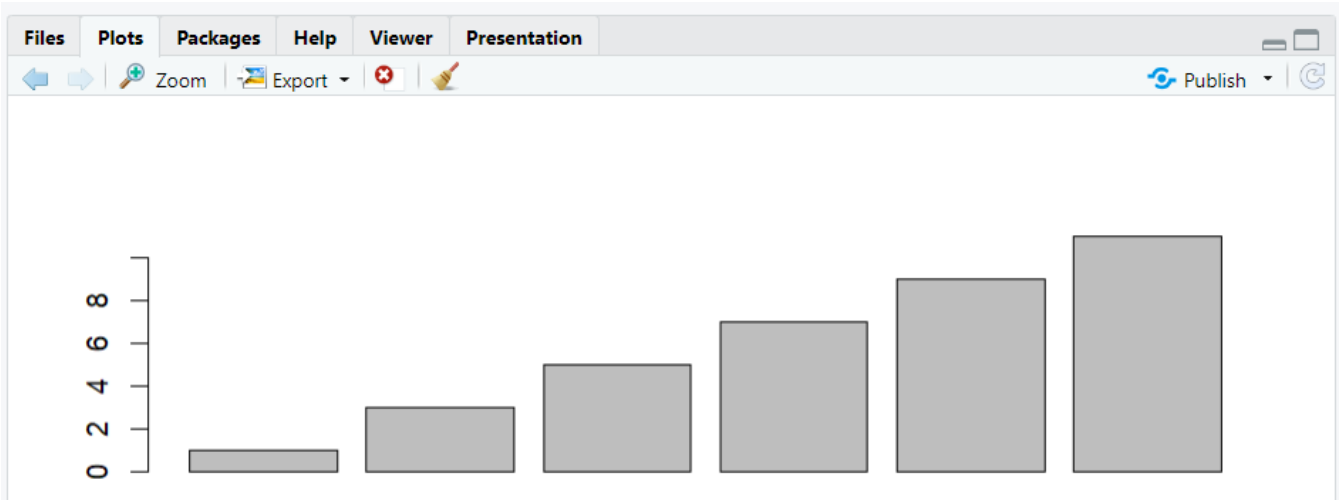
```
Console   Terminal ×   Background Jobs ×
R 4.3.1 · ~/
> labelset <- c('one', 'three', 'five', 'seven', 'nine', 'eleven')
> x1 <- c(1,3,5,7,9,11)
> y1 <- c(2,7,5,10,8,10)
> plot(x1,y1, type='o', lty=3, col='pink', lwd=4, col.main="blue", xlab="Time", y
lab="Performance", main="Plot")
> text(x1+0.5,y1,labelset,col="red")
```



Comandi di Alto Livello

barplot(x)	Produce un grafico a barre dei valori presenti nel vettore x sulla base delle frequenze.
boxplot(x)	Produce un boxplot dei valori presenti nel vettore x sulla base delle frequenze.
boxplot(split(x, f))	Se x è un vettore e f un fattore, la funzione realizza nella stes- sa immagine dei boxplot relativi alla variabile x per ciascuna delle modalità indicate nel fattore f.

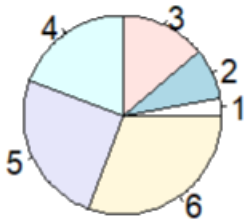




Comandi di Alto Livello

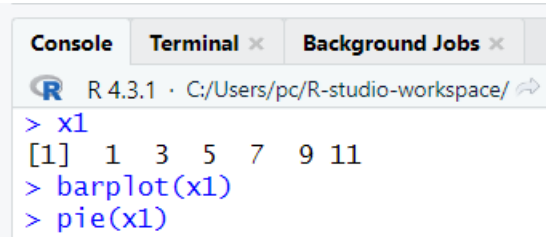
pie(x)	Fornisce un grafico a torta dei valori presenti in x sulla base delle frequenze.

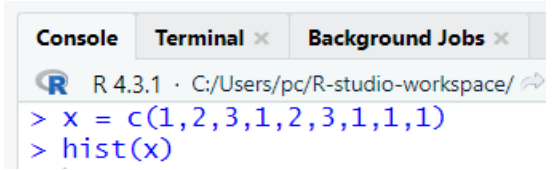
```
Console   Terminal ×   Background Jobs ×
R 4.3.1 · C:/Users/pc/R-studio-workspace/
> x1
[1]  1  3  5  7  9 11
> barplot(x1)
> pie(x1)
```

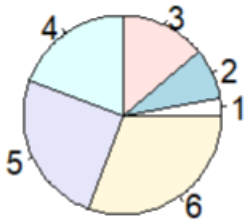


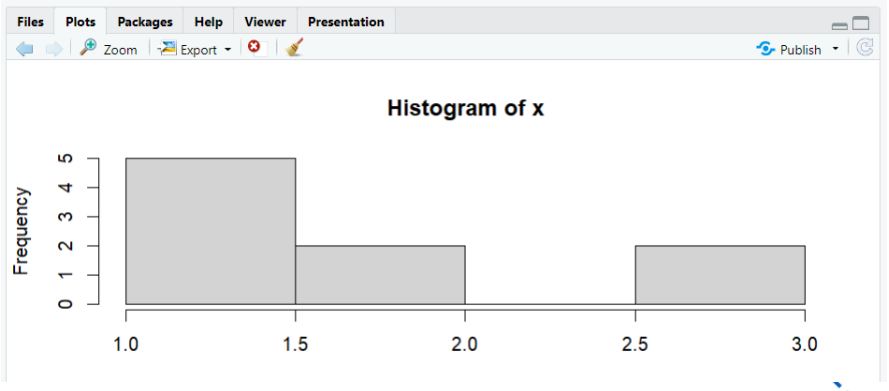
Comandi di Alto Livello

pie(x)	Fornisce un grafico a torta dei valori presenti in x sulla base delle frequenze.
hist(x)	Produce l'istogramma sulla base delle frequenze calcolate su x. Il numero di classi viene scelto per default.
hist(x, nclass = n)	Produce l'istogramma sulla base delle frequenze calcolate su x e viene indicato il numero di classi.
hist(x, breaks = ...)	Produce l'istogramma sulla base delle frequenze calcolate su x e sono indicati gli estremi degli intervalli.





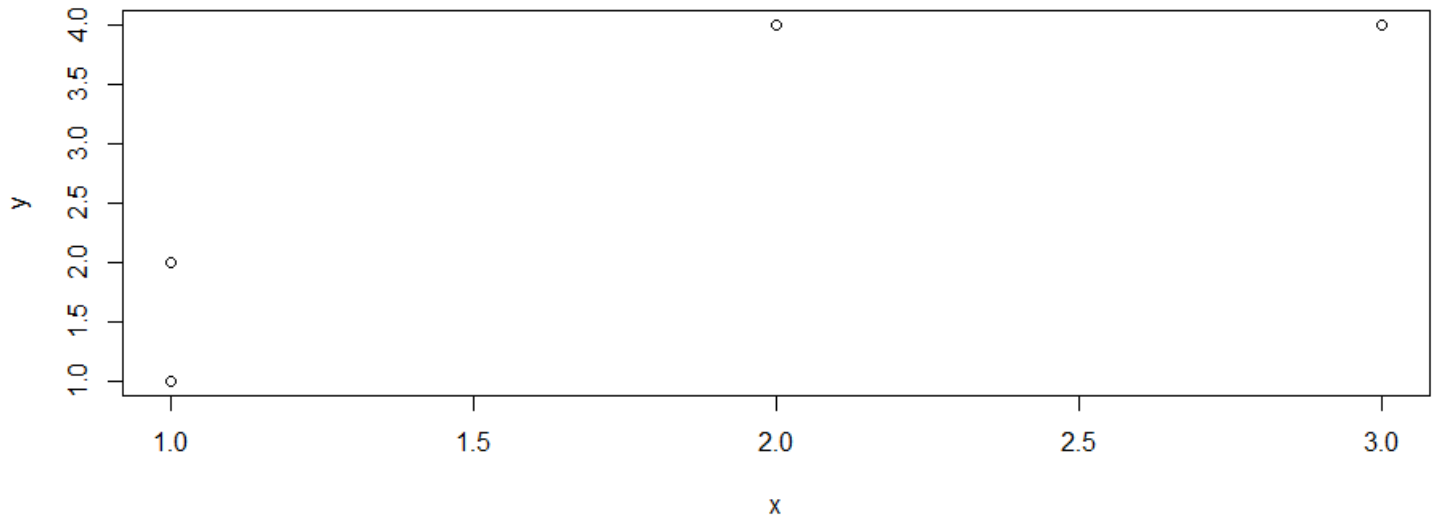




Comandi di Alto Livello

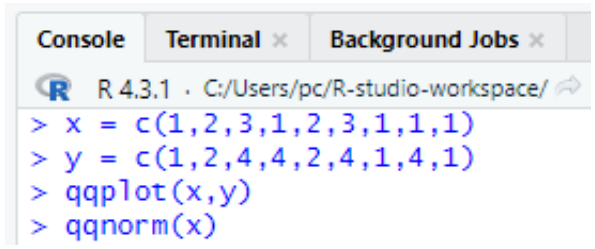
qqplot(x, y)	Confronta le distribuzioni empiriche di x e di y, ossia produce un grafico dei quantili dei dati del vettore x rispetto ai quantili dei dati del vettore y.

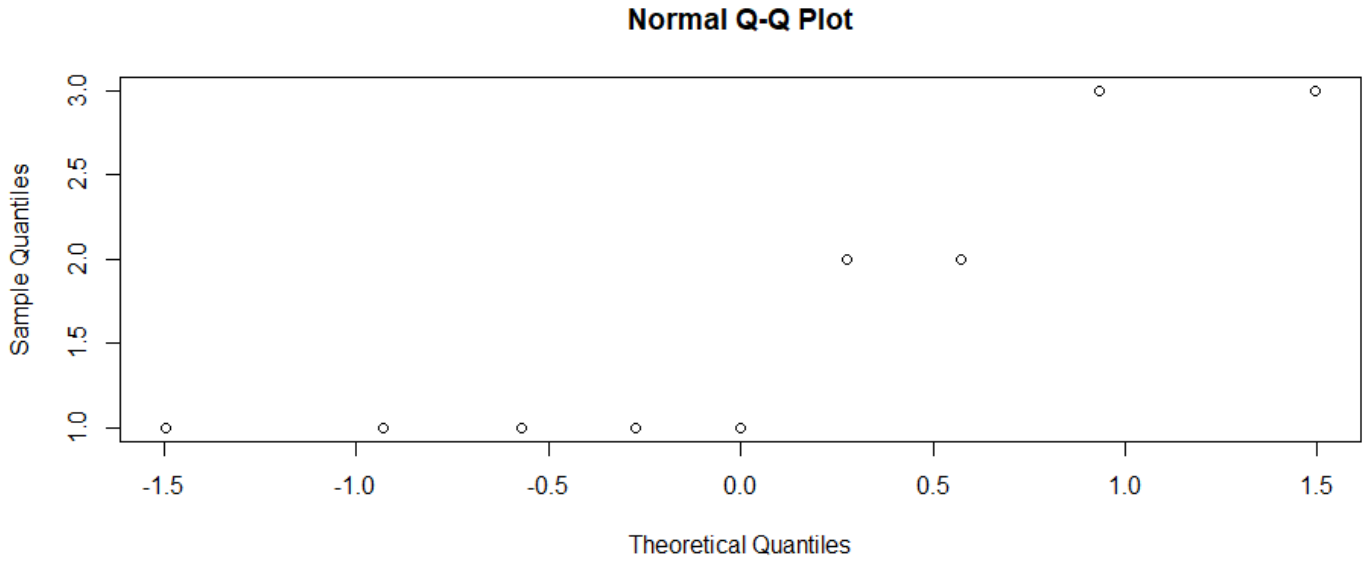
```
Console   Terminal ×   Background Jobs ×
R 4.3.1 · C:/Users/pc/R-studio-workspace/
> x = c(1,2,3,1,2,3,1,1,1)
> y = c(1,2,4,4,2,4,1,4,1)
> qqplot(x,y)
```



Comandi di Alto Livello

qqplot(x, y)	Confronta le distribuzioni empiriche di x e di y, ossia produce un grafico dei quantili dei dati del vettore x rispetto ai quantili dei dati del vettore y.
qqnorm(x)	Produce un grafico quantile-quantile dei dati del vettore x rispetto ad una corrispondente distribuzione normale standard.

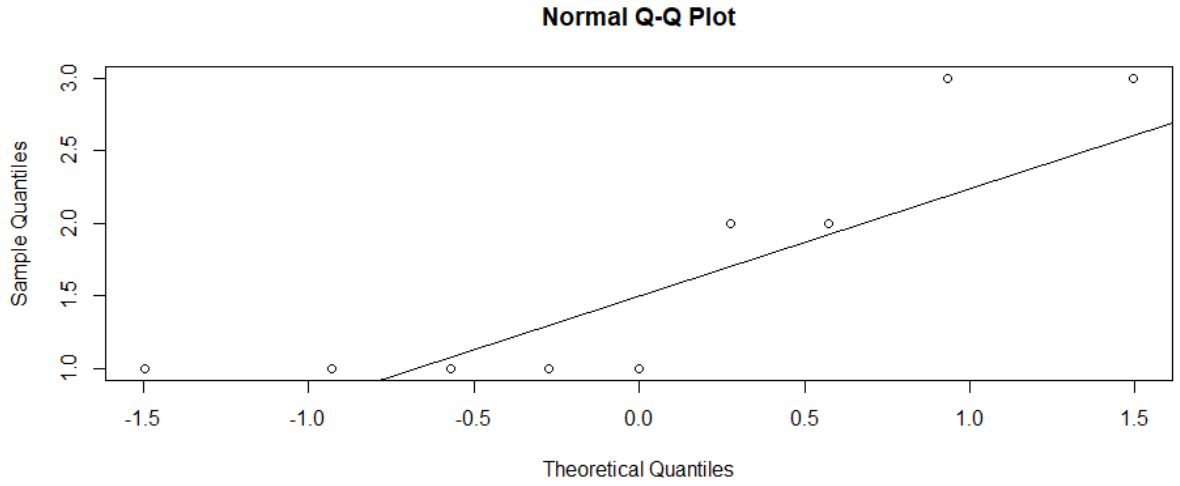




Comandi di Alto Livello

qqplot(x, y)	Confronta le distribuzioni empiriche di x e di y, ossia produce un grafico dei quantili dei dati del vettore x rispetto ai quantili dei dati del vettore y.
qqnorm(x)	Produce un grafico quantile-quantile dei dati del vettore x rispetto ad una corrispondente distribuzione normale standard.
qqline(x)	Funzione di basso livello che aggiunge una linea che passa attraverso il primo e il terzo quartile.

```
Console   Terminal ×   Background Jobs ×
R 4.3.1 · C:/Users/pc/R-studio-workspace/
> x = c(1,2,3,1,2,3,1,1,1)
> y = c(1,2,4,4,2,4,1,4,1)
> qqplot(x,y)
> qqnorm(x)
> qqline(x)
```

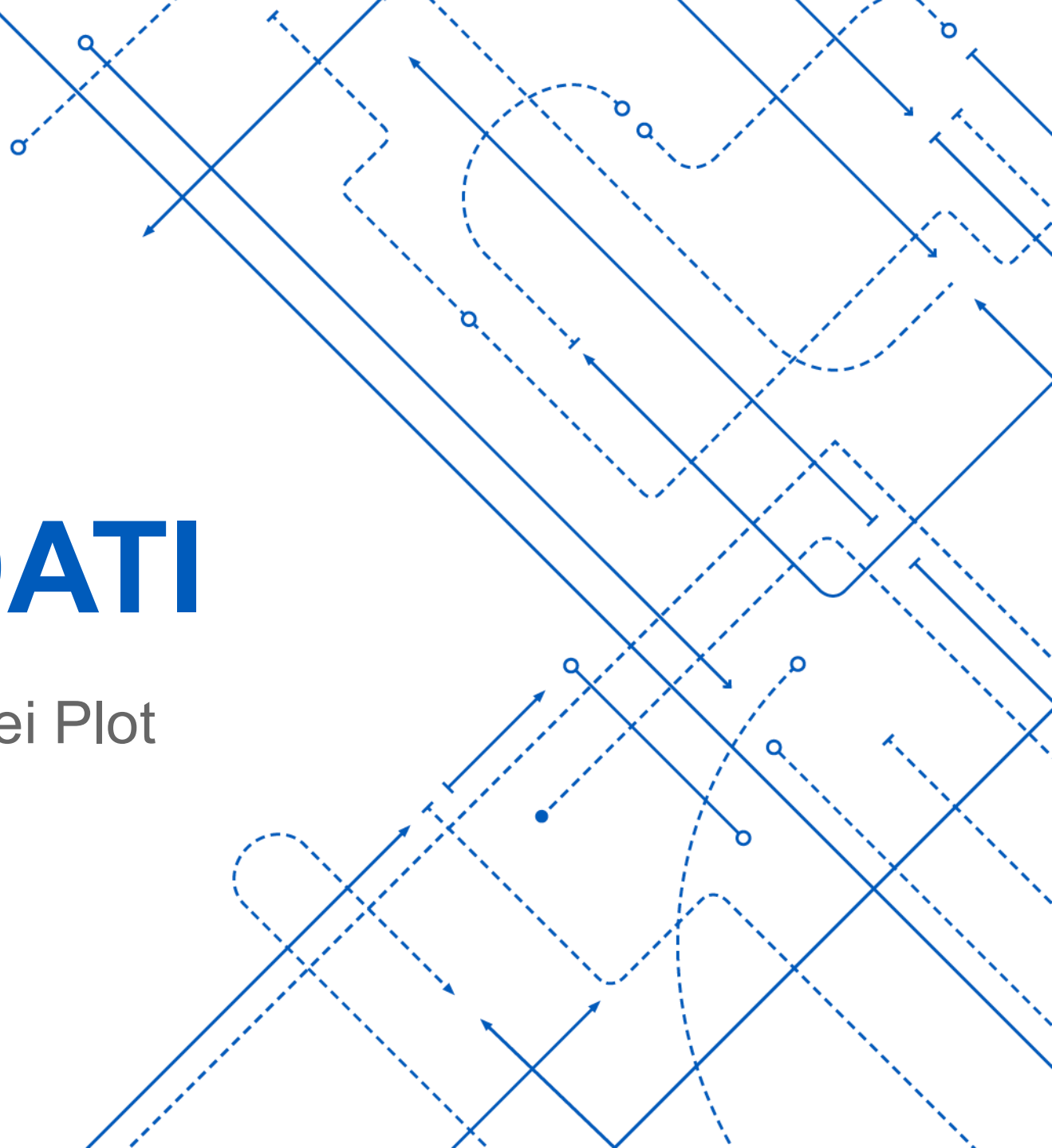


STATISTICA E ANALISI DEI DATI

Capitolo 3 – Personalizzazione dei Plot

Dott. Stefano Cirillo
Dott. Luigi Di Biasi

a.a. 2023-2024



Alcuni Parametri Aggiuntivi

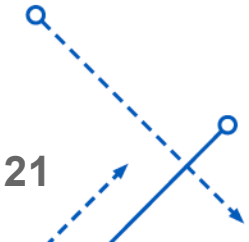
Area =	Se si utilizza FALSE non vengono disegnati gli assi e il contorno
col =	Colore usato per evidenziare le linee, i punti e i rettangoli; si può usare anche un vettore di colori per evidenziare colori multipli.
lty =	Tipo di linea richiesto (0 = tratto bianco, 1 = linea continua, 2 =linea tratteggiata, 3 = linea punteggiata, 4 = linea punto-tratto, 5 = linea con tratto lungo, 6 = linea con doppio tratto).
lwd =	Spessore delle linee.
log = ”x” log = ”y” log = ”xy”	Indica di porre in scala logaritmica la variabile posta sull’asse delle ascisse o delle ordinate o entrambe
type = ”x”	Il carattere ”**x**” indica il tipo di grafico da visualizzare; con ”**p**” si ha una nuvola di punti individuali, con ”**l**” delle linee connesse, con ”b” punti connessi da linee, con ”**o**” punti connessi da linee che passano sopra i punti, con ”**h**” linee verticali dai punti dell’asse delle ascisse, con ”**s**” o ”**S**” una funzione a gradini, con ”n” nessun grafico.
xlab = ylab =	”**str**” Stringa da associare all’asse delle ascisse. ”**str**” Stringa da associare all’asse delle ordinate.
xlim = c(n1, n2) ylim = c(n1, n2)	Limite inferiore n1 e superiore n2 dell’asse delle ascisse. Limite inferiore n1 e superiore n2 dell’asse delle ordinate.
border =	Permette di indicare il colore da utilizzare per colorare il contorno di figure come rettangoli o poligoni.

Alcune funzioni grafiche di basso livello

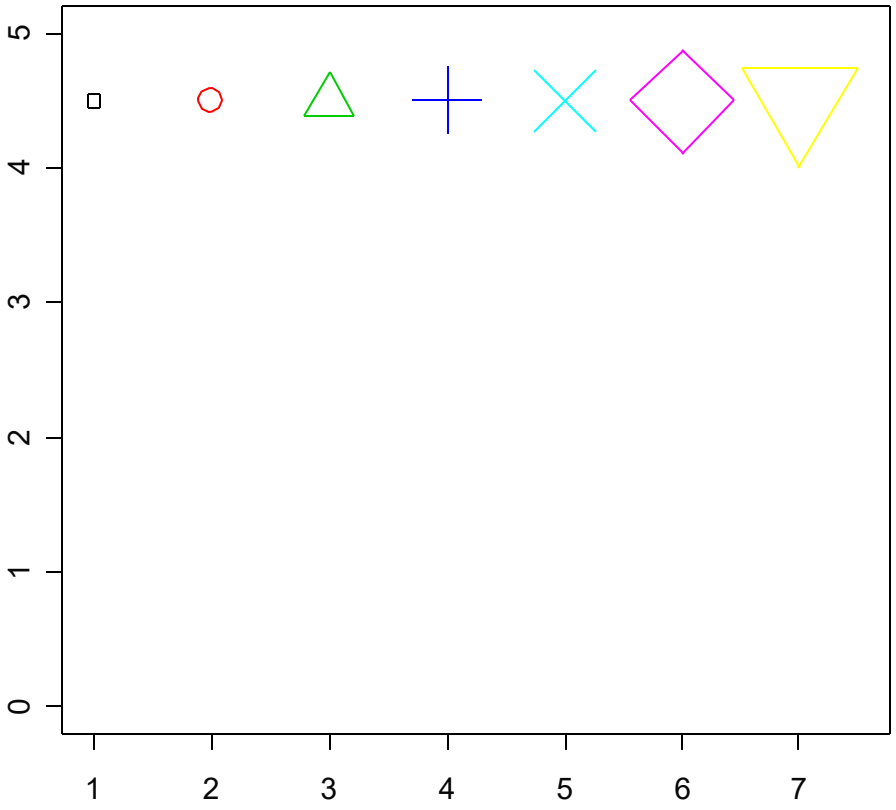
- È possibile modificare un grafico generato con funzioni ad alto livello con funzioni di basso livello, che possono aggiungere elememto come:
  - Punti,
  - Linee
  - Testo
  - Etc.

- Le coordinate sono fornite in termini di *coordinate utente*, definite da precedenti comandi di alto livello

points(x,y)	Aggiunge un punto al grafico corrente (in posizione x,y)
lines(x,y)	Aggiunge una linea al grafico corrente
text(x,y,label)	Aggiunge la stringa di testo label in posizione x,y
abline(a,b)	Aggiunge una linea di inclinazione a ed intercetta b
polygon(x,y,z, ...)	Disegna un poligono i cui vertici (ordinati) sono elencati come argomenti
legend(x,y,legend)	Aggiunge una legenda in posizione x,y
title(main.sub)	Aggiunge il titolo main ed opzionalmente un sottotitolo sub
axis(side,...)	Aggiunge gli assi nelle posizioni specificate da side



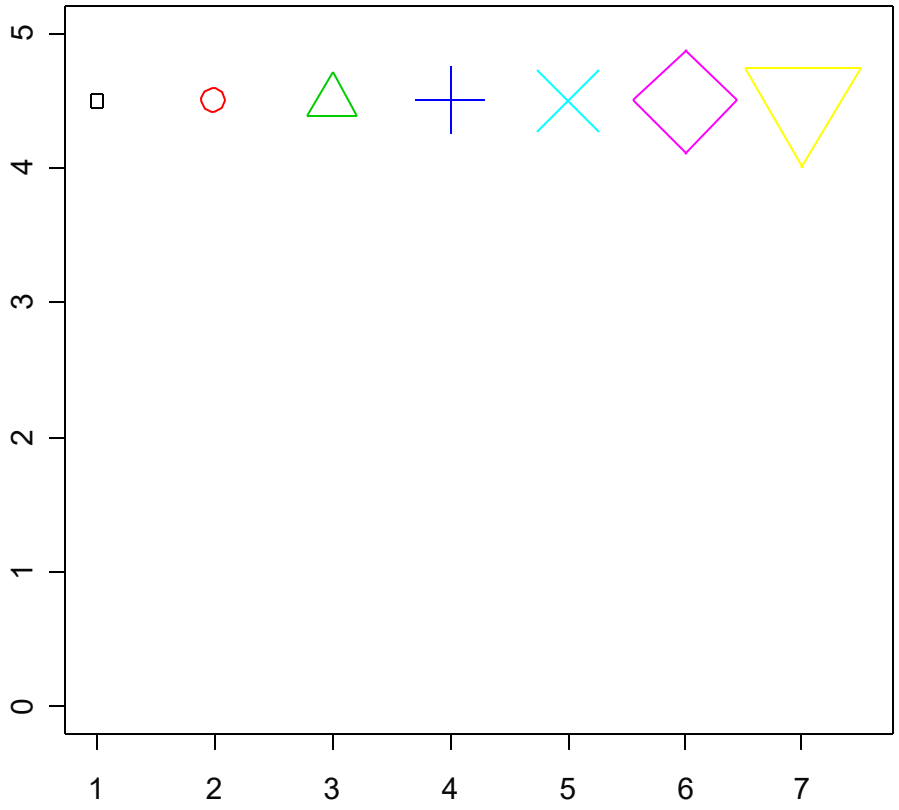
Esempio di funzioni grafiche di basso livello




```
> plot(1, 1, xlim=c(1, 7.5), ylim=c(0,5), type="n")
> points(1:7, rep(4.5, 7), cex=1:7, col=1:7, pch=0:6)
```

- Il parametro **type="n"** vienene utilizzato per creare un grafico vuoto, ovvero imposta il tipo di grafico da tracciare come "nessuno"
- Viene tracciato solo il sistema di coordinate (assi e griglia, se abilitata), senza visualizzare punti, linee o simboli

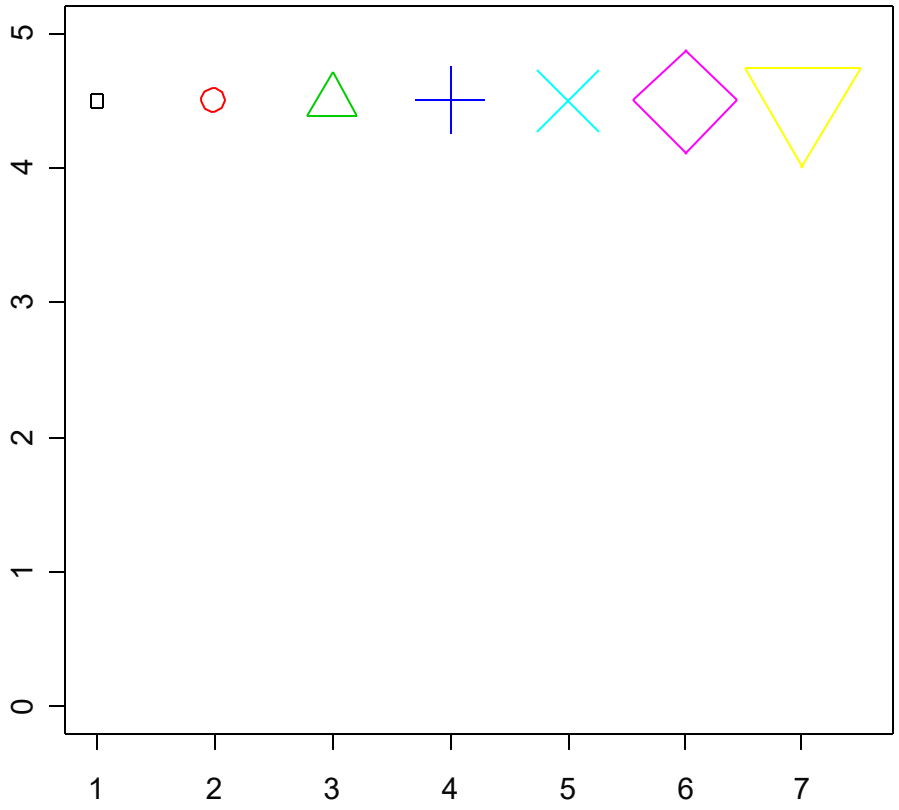
Esempio di funzioni grafiche di basso livello




```
> plot(1, 1, xlim=c(1, 7.5), ylim=c(0,5), type="n")
> points(1:7, rep(4.5, 7), cex=1:7, col=1:7, pch=0:6)
```

- Il parametro **cex** nel comando points() in R viene utilizzato per specificare la dimensione dei punti che vengono tracciati nel grafico:
  - Un valore di cex = 1 indica la dimensione predefinita
  - Valori maggiori di 1 ingrandiscono i punti (ad esempio, cex = 1.5 rende i punti il 50% più grandi)
  - Valori inferiori a 1 riducono le dimensioni dei punti (ad esempio, cex = 0.5 rende i punti la metà della dimensione predefinita)

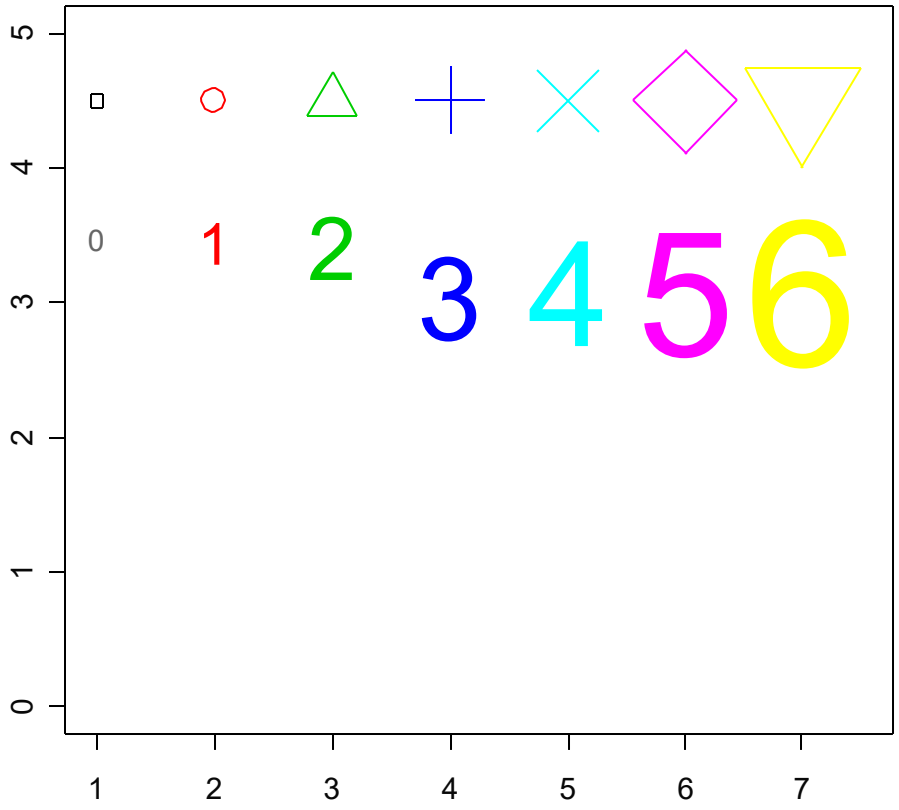
Esempio di funzioni grafiche di basso livello




```
> plot(1, 1, xlim=c(1, 7.5), ylim=c(0,5), type="n")
> points(1:7, rep(4.5, 7), cex=1:7, col=1:7, pch=0:6)
```

- Il parametro **pch** nel comando points() in R controlla il tipo di simbolo utilizzato per tracciare i punti nel grafico
  - Puoi specificare diversi simboli utilizzando i valori numerici da 0 a 25 o fornendo un singolo carattere
    - pch = 0: quadrato vuoto
    - pch = 1: cerchio vuoto
    - pch = 2: triangolo vuoto rivolto verso l'alto
    - pch = 3: croce
    - pch = 4: croce diagonale (X)
    - pch = 5: diamante vuoto
    - pch = 6: triangolo vuoto rivolto verso il basso
    - …

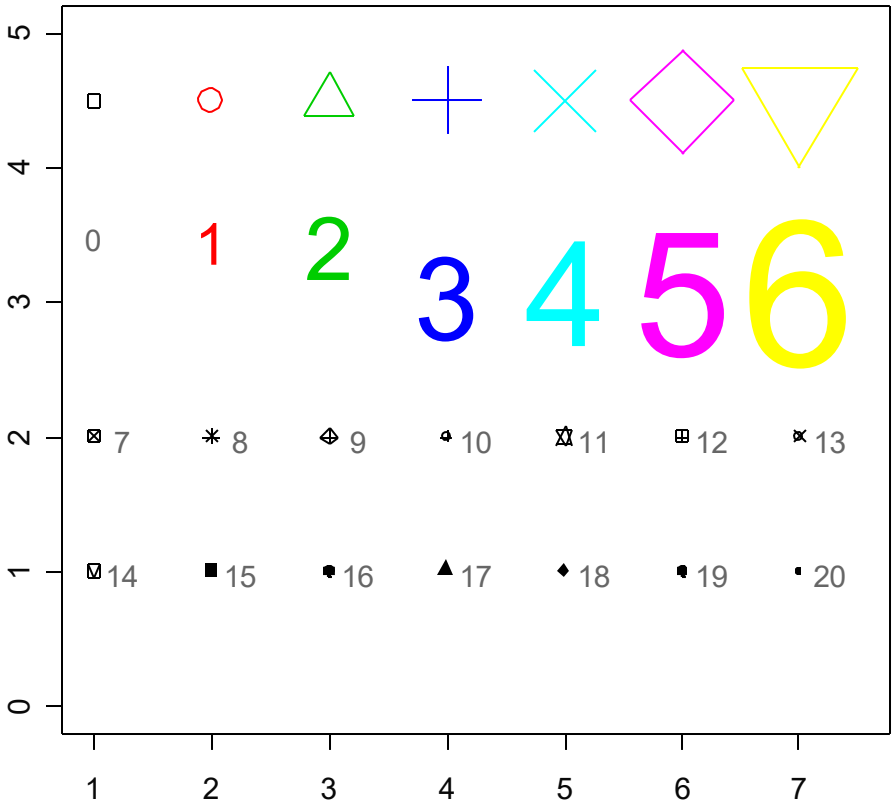
Esempio di funzioni grafiche di basso livello




```
> plot(1, 1, xlim=c(1, 7.5), ylim=c(0,5), type="n")
> points(1:7, rep(4.5, 7), cex=1:7, col=1:7, pch=0:6)
> text(1:7,rep(3.5, 7), labels=paste(0:6), cex=1:7, col=1:7)
```

- **labels = paste(0:6)**: Crea le etichette da visualizzare. paste(0:6) genera le etichette "0", "1", "2", "3", "4", "5", "6", che saranno i numeri visualizzati nei punti specificati
- **cex = 1:7**: Imposta la dimensione del testo in modo crescente
    - Il testo al primo punto avrà dimensione normale (cex = 1), mentre il testo al settimo punto avrà dimensione maggiore (cex = 7)

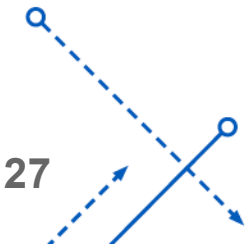
Esempio di funzioni grafiche di basso livello




```
> plot(1, 1, xlim=c(1, 7.5), ylim=c(0,5), type="n")
> points(1:7, rep(4.5, 7), cex=1:7, col=1:7, pch=0:6)
> text(1:7,rep(3.5, 7), labels=paste(0:6), cex=1:7, col=1:7)
> points(1:7,rep(2,7), pch=(0:6)+7)
> text((1:7)+0.25, rep(2,7), paste((0:6)+7))
> points(1:7,rep(1,7), pch=(0:6)+14)
> text((1:7)+0.25, rep(1,7),paste((0:6)+14))
```

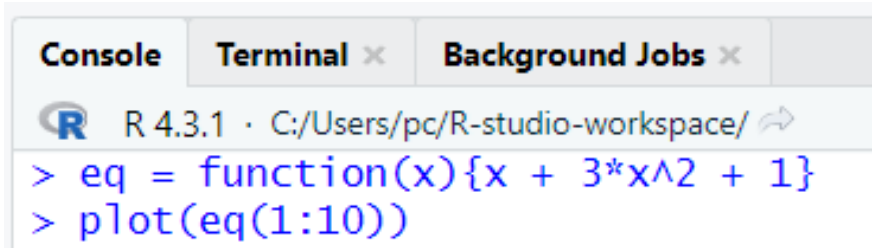
Formule Matematiche nei Grafici

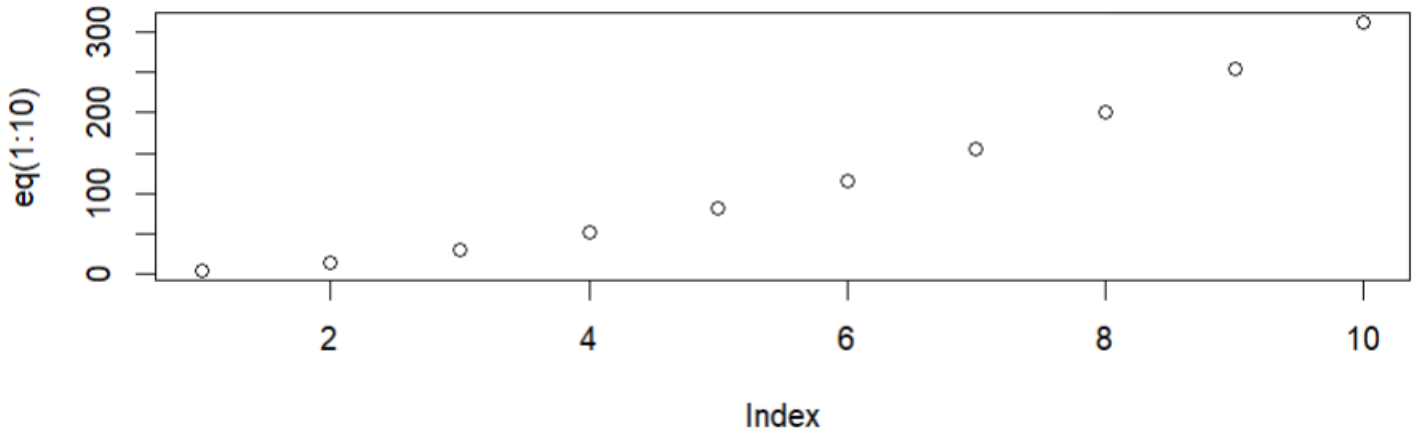
- E’ possibile aggiungere facilmente ad un grafico sia simboli matematici sia delle formule inserendo come parametro all’interno della funzione **expression**() un’espressione che utilizzi una sintassi simile a quella di LATEX
  - l’espressione viene tradotta dall’interprete nella corrispondente espressione matematica rispettando lo stile utilizzato nei documenti LATEX
    - **expression**() può essere fornita in ingresso alle funzioni:
      - **text**(x, y, labels=‘Testo’)
        - Aggiunge il testo indicato nel parametro labels nella posizione indicata dalle coordinate (x, y);
      - **mtext**(‘Testo’, side)
        - Aggiunge il testo indicato sul margine scelto con il parametro side (1,2,3,4);
      - **axis**(side, …):
        - Aggiunge un asse nel lato indicato dall’intero side: 1 (basso), 2 (sinistra), 3 (alto), 4 (destra);
      - **title**(‘Titolo’, ’Sottotitolo’):
        - Aggiunge il titolo (sopra) e anche un sottotitolo (sotto) al grafico.



Formule Matematiche nei Grafici

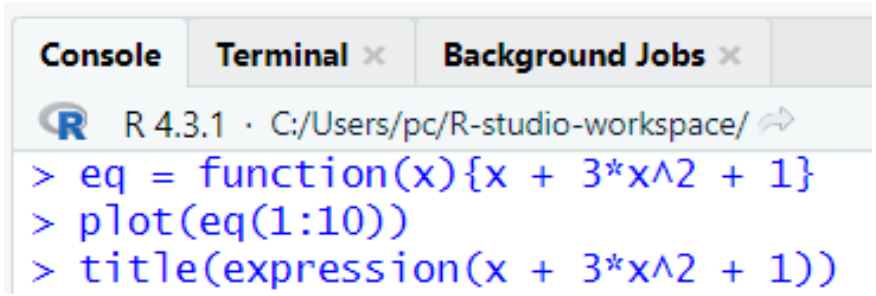
- E’ possibile aggiungere facilmente ad un grafico sia simboli matematici sia delle formule inserendo come parametro all’interno della funzione **expression**() un’espressione che utilizzi una sintassi simile a quella di LATEX
  - l’espressione viene tradotta dall’interprete nella corrispondente espressione matematica rispettando lo stile utilizzato nei documenti LATEX

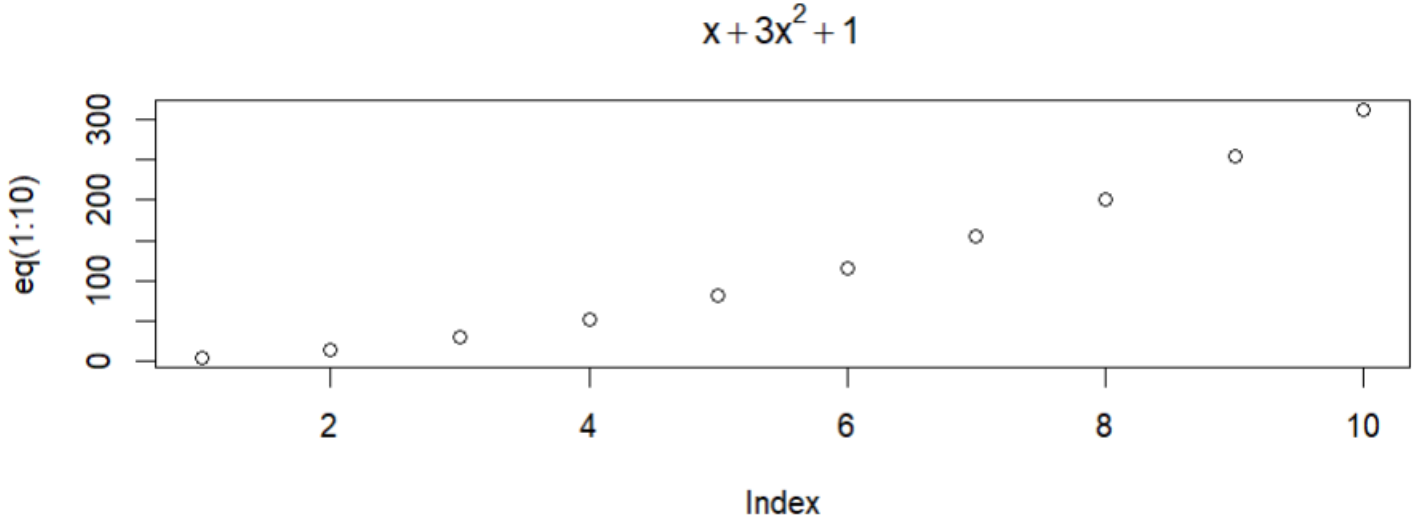




Formule Matematiche nei Grafici

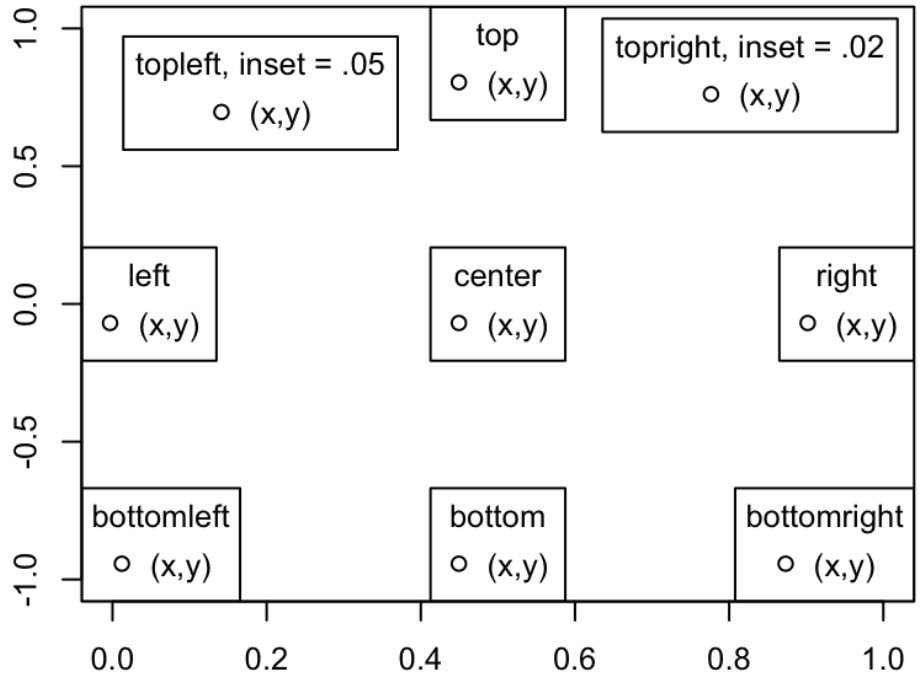
- E’ possibile aggiungere facilmente ad un grafico sia simboli matematici sia delle formule inserendo come parametro all’interno della funzione **expression**() un’espressione che utilizzi una sintassi simile a quella di LATEX
  - l’espressione viene tradotta dall’interprete nella corrispondente espressione matematica rispettando lo stile utilizzato nei documenti LATEX





LEGENDA

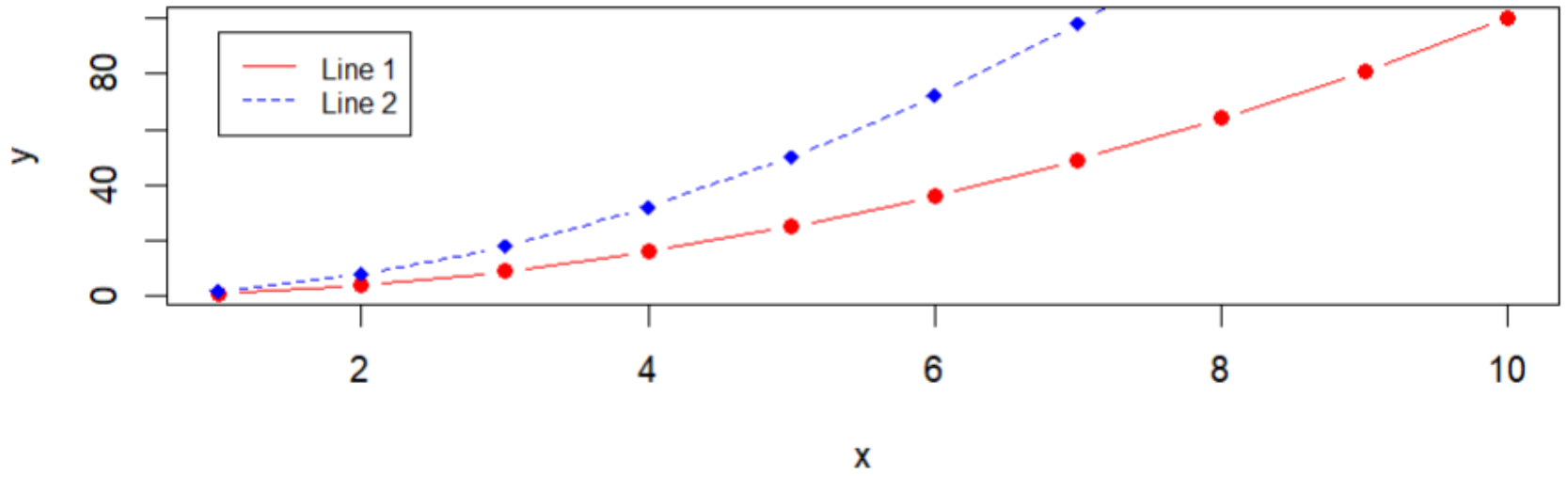
- Per aggiungere la legenda all'interno di un grafico si utilizza la fuzione **legend**() con la quale le componenti dei vettori vengono affiancate nel momento della creazione del testo
  - **legend**(x, y, legend, . . .): Aggiunge una legenda al disegno nel punto (x, y)
  - Occorre fornire un **vettore v** della stessa lunghezza di legend e impostare i parametri di tale vettore.
- La posizione può essere specificata usando:
  - Bottomright
  - Bottom
  - Bottomleft
  - Left
  - Topleft
  - Top
  - ….



LEGENDA

- Per aggiungere la legenda all'interno di un grafico si utilizza la fuzione **legend**() con la quale le componenti dei vettori vengono affiancate nel momento della creazione del testo
  - **legend**(x, y, legend, . . .): Aggiunge una legenda al disegno nel punto (x, y)

```
> x <- 1:10;
> y1 = x*x;
> y2 = 2*y1
> plot(x, y1, type="b", pch=19, col="red", xlab="x", ylab="y")
> lines(x, y2, pch=18, col="blue", type="b", lty=2)
> legend(1, 95, legend=c("Line 1", "Line 2"), col=c("red", "blue"), lty=1:2, cex=0.8)
>
```

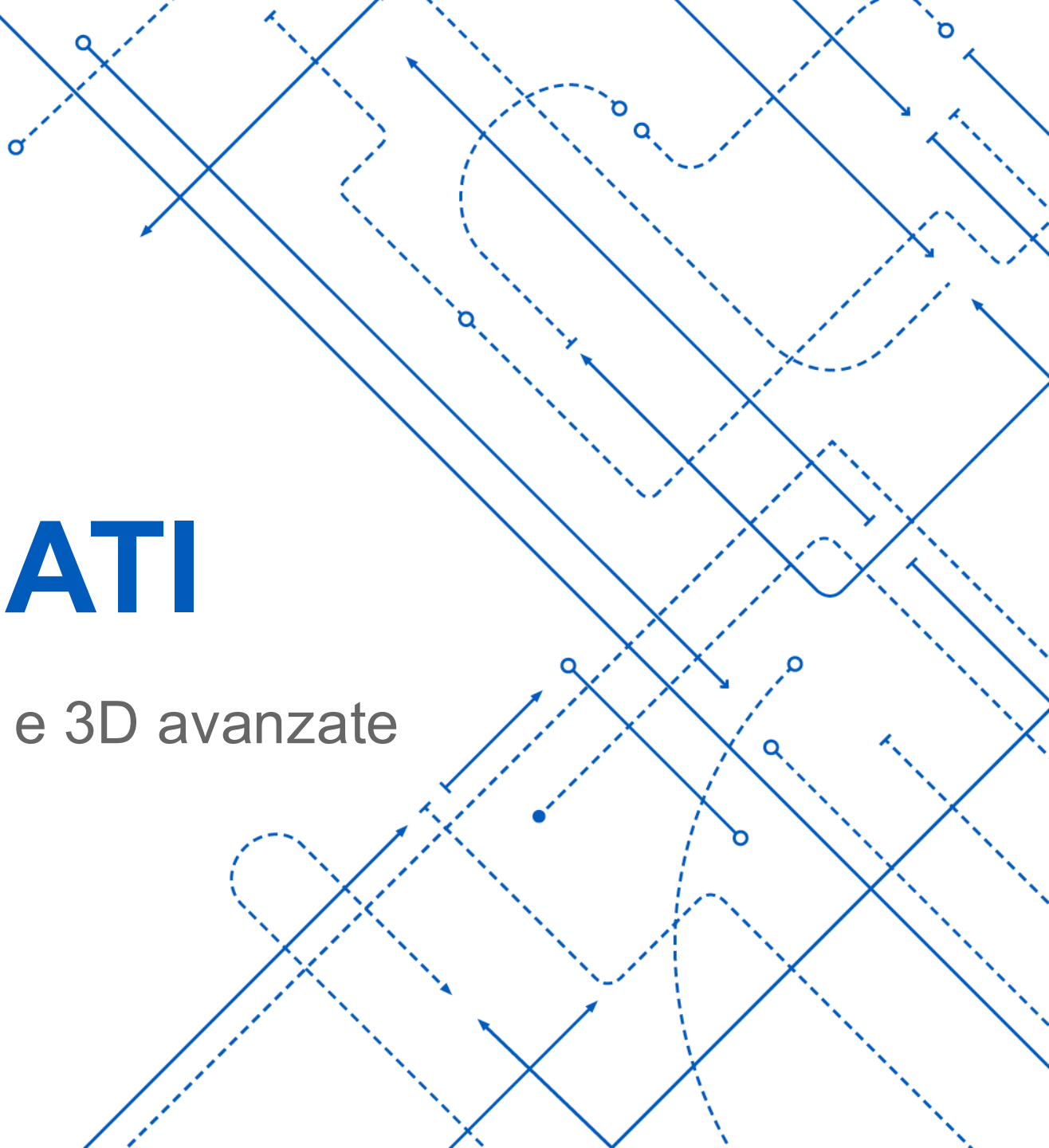


STATISTICA E ANALISI DEI DATI

Capitolo 3 - Rappresentazioni 2D e 3D avanzate

Dott. Stefano Cirillo
Dott. Luigi Di Biasi

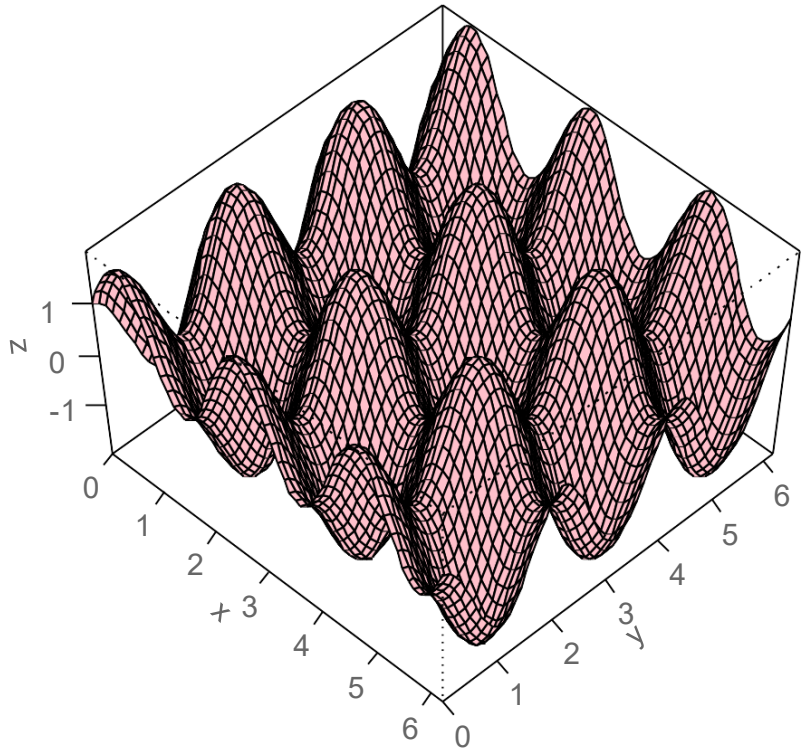
a.a. 2023-2024



FUNZIONE PERSP()

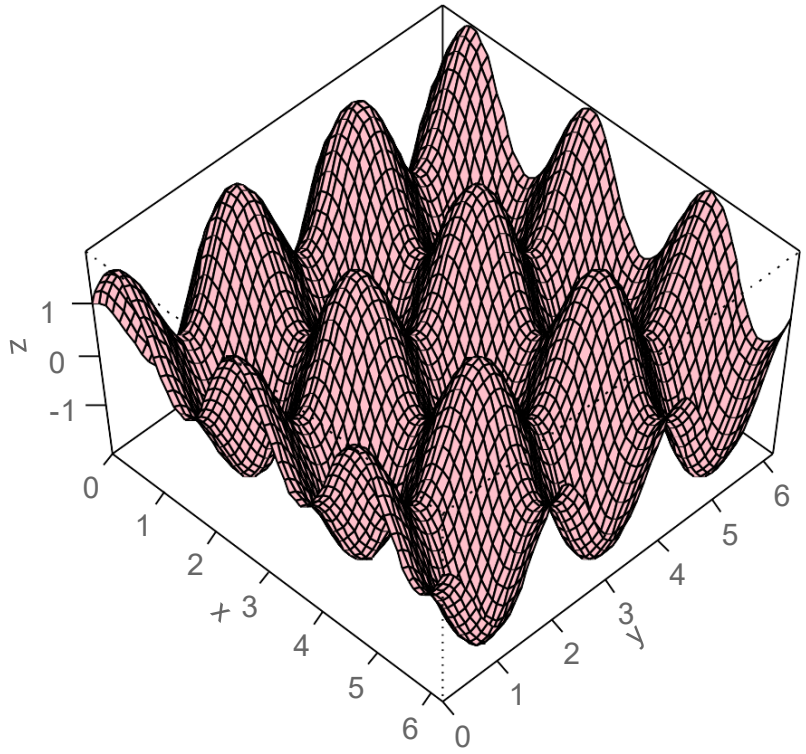
- La funzione persp() in R viene utilizzata per creare grafici tridimensionali a griglia, che mostrano una superficie 3D
- Parametri di persp():

1. **x, y, z**:
    1) **x**: vettore dei valori sull'asse x.
    2) **y**: vettore dei valori sull'asse y.
    3) **z**: matrice di valori che rappresenta la superficie, dove ogni elemento z[i, j] è il valore della funzione in corrispondenza di (x[i], y[j]).
2. **theta**: Angolo di rotazione della superficie rispetto all'asse z. Il valore predefinito è 0. Modificarlo permette di ruotare la superficie e vedere il grafico da diverse prospettive.
3. **phi**: Angolo di visualizzazione verticale (inclinazione dall'alto o dal basso) rispetto all'orizzontale. Il valore predefinito è 0.
4. **shade**: Controlla l'ombreggiatura sulla superficie per dare una percezione più chiara della profondità. Un valore vicino a 1 aumenta l'ombreggiatura, mentre 0 la disabilita.
5. **border**: Colore dei bordi della griglia. Utilizzato per dare maggiore definizione alla superficie.
6. **expand**: Un fattore di scala per espandere o ridurre la dimensione della superficie rispetto al grafico.
7. **xlab, ylab, zlab**: Etichette per gli assi. Puoi usare espressioni matematiche per scrivere formule o simboli matematici



FUNZIONE PERSP()

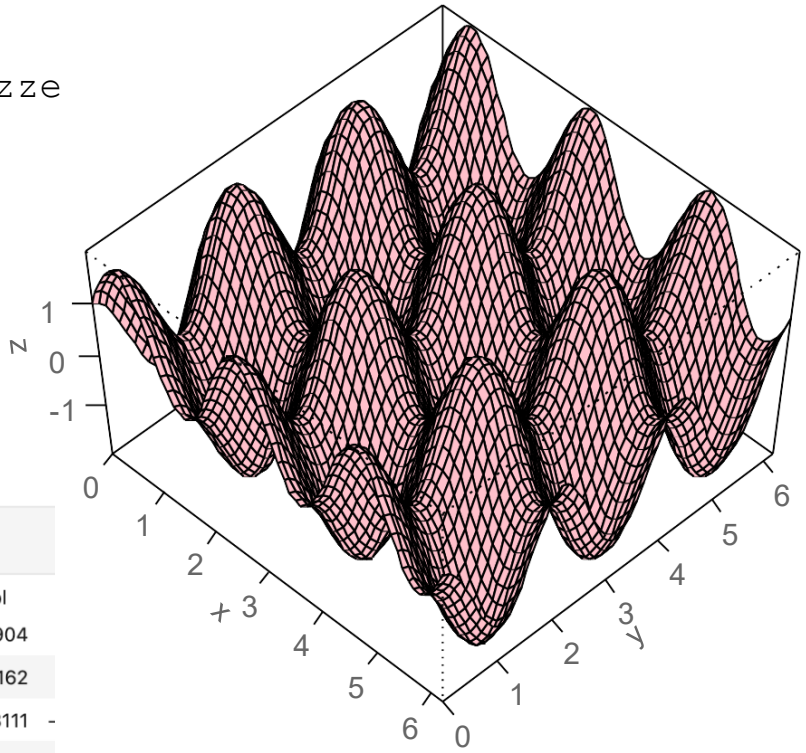
- In ambito di ricerca informatica, la funzione persp() può essere utile per visualizzare dati che dipendono da due variabili o per rappresentare superfici di funzioni matematiche legate a modelli di apprendimento automatico, reti neurali o analisi delle prestazioni di algoritmi
  - Ad esempio, consideriamo un problema legato alla funzione di costo di una rete neurale
    - In un contesto di ottimizzazione, potremmo voler visualizzare come cambia il valore della funzione di costo rispetto a due parametri, per comprendere meglio come ottimizzare un modello
  - Ad esempio, può essere utile per il monitoraggio dei processi evolutivi di un algoritmo euristico/evoluzionistico
    - Ad esempio per capire se un algoritmo si 'ferma' in un ottimo locale
  - …



FUNZIONE PERSP()

```
> x<-y<-seq(0,2*pi,by=0.1)
> z <- outer(sin(3*x),cos(3*y),"+") # crea la matrice z delle altezze
> persp(x,y,z,phi=60,theta=45,d=10,col="pink",ticktype="detailed")
```

- Il comando z <- outer(sin(3*x), cos(3*y), "+") crea una matrice z combinando gli elementi dei vettori sin(3*x) e cos(3*y) tramite l'operazione indicata, in questo caso l'addizione ("+")



Rappresentazione della funzione:

f(x,y) = sin(3x) + cos(3y)

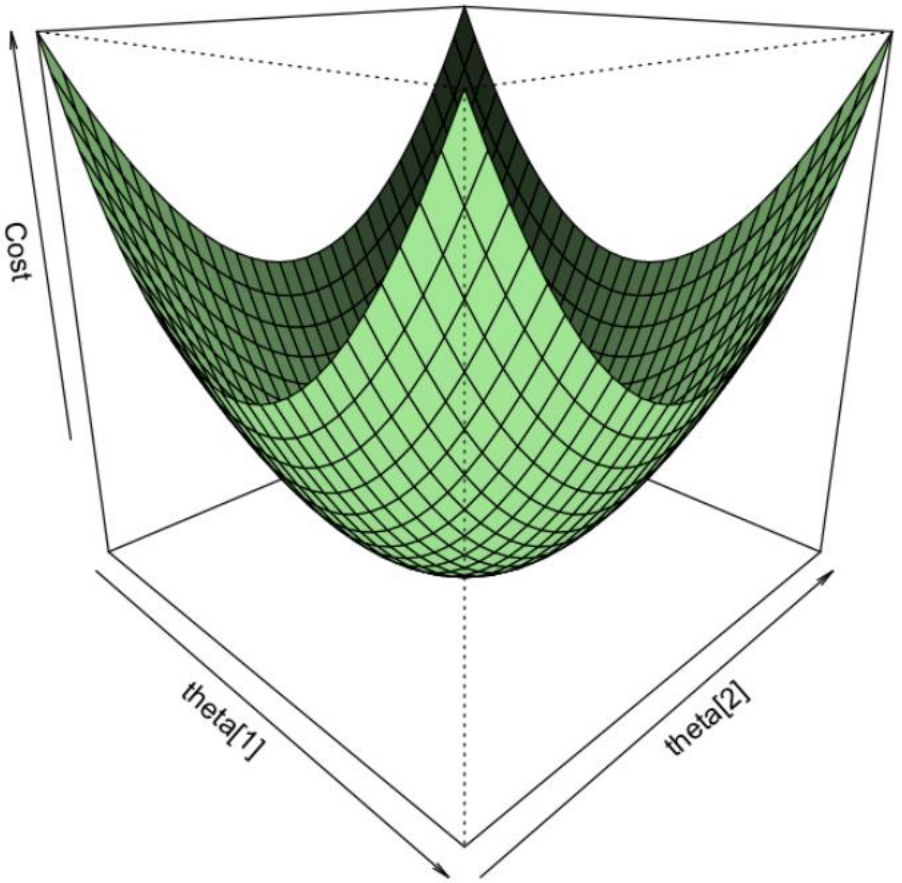
```
z <- outer(sin(3*x),cos(3*y),"+")
z
```

A matrix: 30 × 30 of type dbl

1.14228307	0.04638240	1.73365417	1.21713703	0.023467680	1.68069834	1.29065825	0.006163980	1.62371316	1.36241904	⋯	1.36241904
-0.18234435	-1.27824502	0.40902675	-0.10749039	-1.301159740	0.35607092	-0.03396917	-1.318463440	0.29908574	0.03779162	⋯	0.03779162
-0.51211708	-1.60801776	0.07925401	-0.43726312	-1.630932475	0.02629819	-0.36374191	-1.648236175	-0.03068699	-0.29198111	⋯	-0.29198111
1.12765307	0.03175239	1.71902416	1.20250703	0.008837675	1.66606834	1.27602824	-0.008466025	1.60908316	1.34778904	⋯	1.34778904
-0.10959775	-1.20549842	0.48177335	-0.03474379	-1.228413139	0.42881752	0.03877743	-1.245716839	0.37183235	0.11053822	⋯	0.11053822
-0.56700635	-1.66290703	0.02436474	-0.49215240	-1.685821748	-0.02859109	-0.41863118	-1.703125448	-0.08557626	-0.34687039	⋯	-0.34687039
1.10736064	0.01145996	1.69873173	1.18221460	-0.011454755	1.64577591	1.25573581	-0.028758455	1.58879073	1.32749661	⋯	1.32749661
-0.03531629	-1.13121697	0.55605480	0.03953766	-1.154131688	0.50309897	0.11305888	-1.171435388	0.44611380	0.18481967	⋯	0.18481967
-0.61769996	-1.71360064	-0.02632886	-0.54284600	-1.736515354	-0.07928469	-0.46932479	-1.753819054	-0.13626987	-0.39756399	⋯	-0.39756399
1.08152383	-0.01437685	1.67289492	1.15637779	-0.037291567	1.61993909	1.22989900	-0.054595267	1.56295392	1.30165979	⋯	1.30165979

FUNZIONE PERSP()

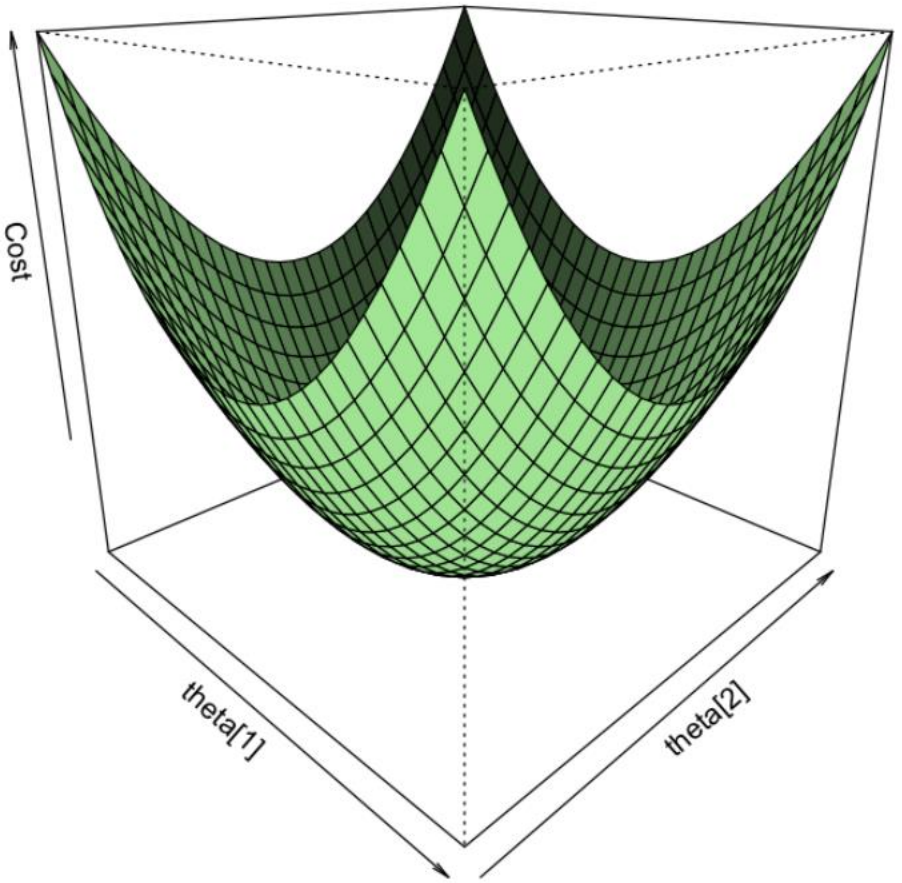
- Visualizzazione della funzione di costo di un modello semplice: Supponiamo di avere una funzione di costo quadratica legata a un problema di regressione lineare
  - Vogliamo visualizzare come cambia il valore della funzione di costo rispetto a due parametri $\vartheta 1$ e $\vartheta 2$, che rappresentano i pesi del modello



FUNZIONE PERSP()

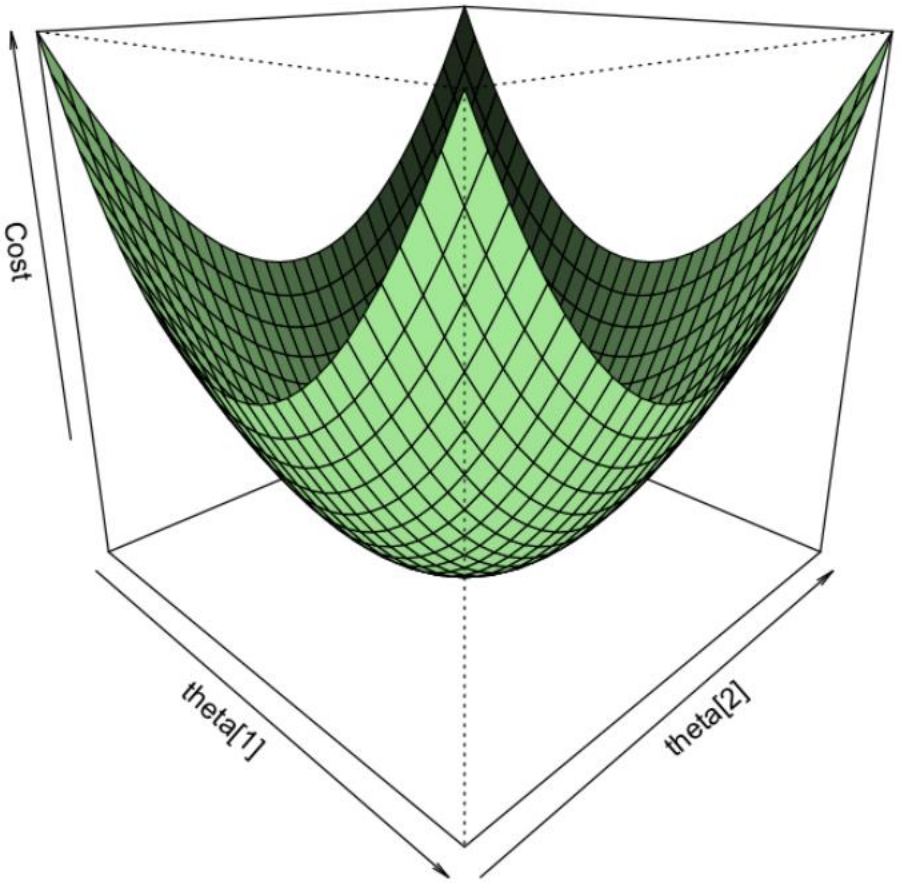
- Visualizzazione della funzione di costo di un modello semplice: Supponiamo di avere una funzione di costo quadratica legata a un problema di regressione lineare
  - Vogliamo visualizzare come cambia il valore della funzione di costo rispetto a due parametri $\vartheta 1$ e $\vartheta 2$, che rappresentano i pesi del modello

```r
# Definisci l'area di theta1 e theta2 (i parametri)
theta1 <- seq(-10, 10, length.out = 30)
theta2 <- seq(-10, 10, length.out = 30)

# Funzione di costo quadratica: cost = theta1^2 + theta2^2 (semplificata)
cost_function <- function(theta1, theta2) {
  return(theta1^2 + theta2^2)  # Funzione di costo quadratica
}

# Crea una griglia di valori (theta1, theta2) per la funzione di costo
cost_values <- outer(theta1, theta2, cost_function)

# Grafico 3D della funzione di costo
persp(theta1, theta2, cost_values,
      theta = 45,        # Angolo di rotazione rispetto all'asse z
      phi = 20,          # Angolo di visualizzazione dall'alto
      col = "lightgreen", # Colore della superficie
      shade = 0.5,       # Ombreggiatura
      border = "black",  # Colore del bordo
      xlab = expression(theta[1]),   # Etichetta asse theta1
      ylab = expression(theta[2]),   # Etichetta asse theta2
      zlab = "Cost")   # Etichetta asse Z (funzione di costo)
```



FUNZIONE PERSP()

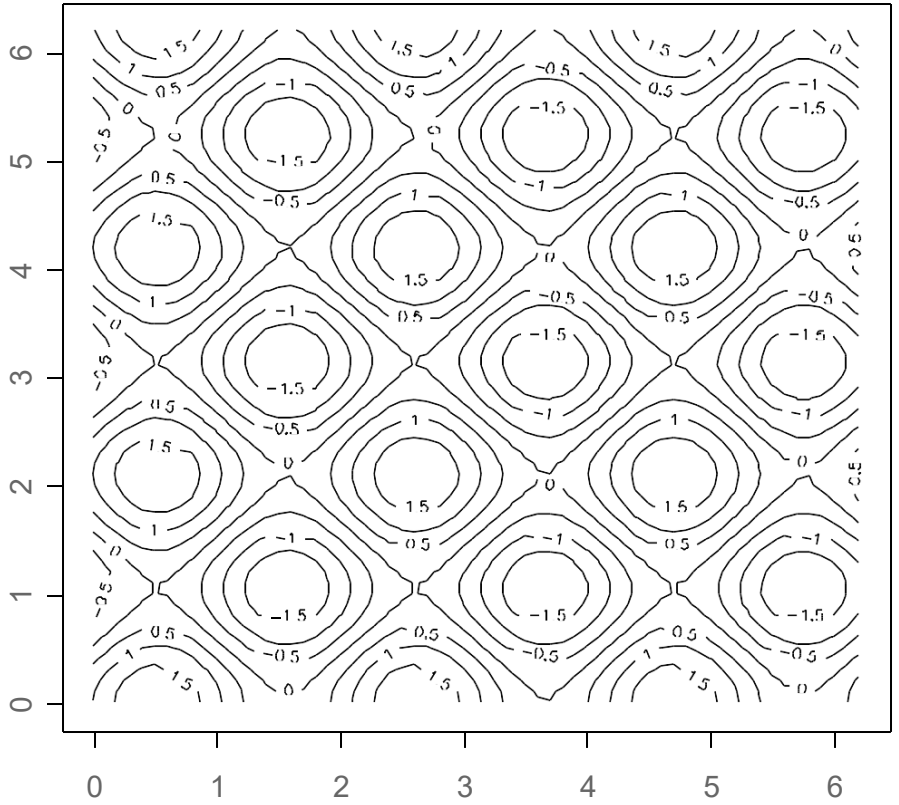
- Visualizzazione della funzione di costo di un modello semplice: Supponiamo di avere una funzione di costo quadratica legata a un problema di regressione lineare
  - Vogliamo visualizzare come cambia il valore della funzione di costo rispetto a due parametri $\vartheta 1$ e $\vartheta 2$, che rappresentano i pesi del modello



- Il grafico mostra una funzione di costo quadratica in cui:
  - I valori più bassi si trovano nel centro (in corrispondenza di $\theta 1 = 0$ e $\theta 2 = 0$)
  - I valori crescono verso l'esterno, rappresentando come la funzione di costo aumenta quando ci si allontana dal minimo ottimo
- Questo tipo di visualizzazione può essere molto utile per capire l'ottimizzazione di modelli di machine learning e come diversi valori dei parametri influenzano la funzione di costo

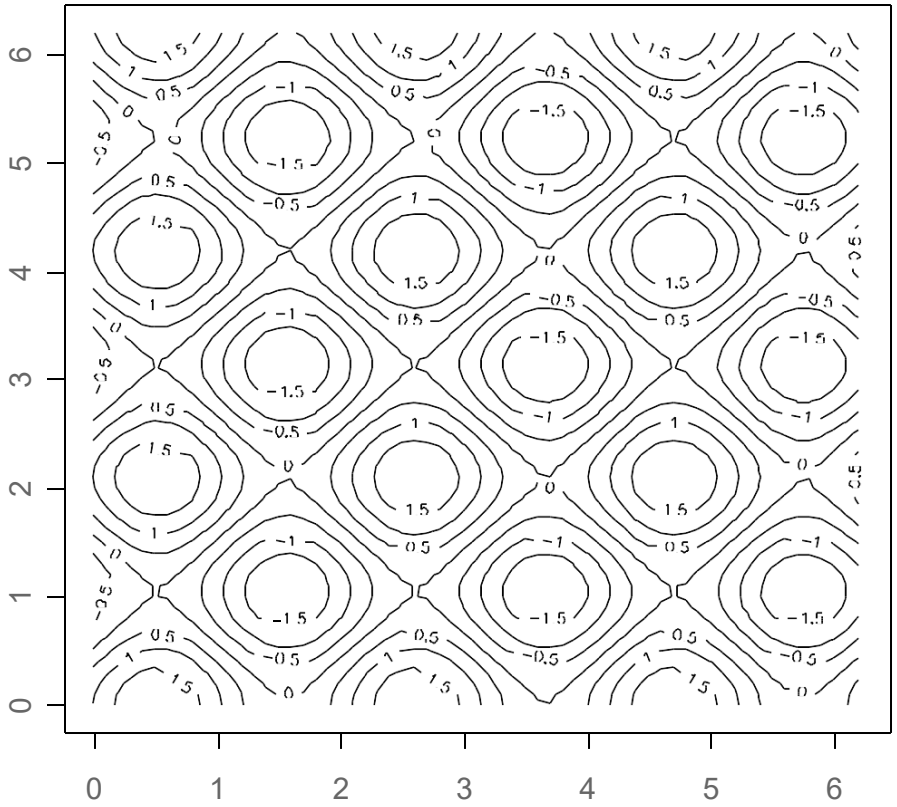
FUNZIONE CONTOUR()

- contour() è una funzione in R utilizzata per disegnare **grafici di curve di livello** (contour plots)
  - Rappresenta una superficie tridimensionale su un piano bidimensionale tramite linee di livello, dove ogni linea rappresenta punti con lo stesso valore z
- Parametri di contour():
  - **x**: vettore di valori per l'asse X.
  - **y**: vettore di valori per l'asse Y.
  - **z**: matrice che rappresenta i valori della superficie z=f(x,y).
  - **levels**: numero o valore specifico dei livelli da visualizzare.
  - **col**: colore delle linee di livello.
  - **lty**: tipo di linea (solida, tratteggiata, ecc.).
  - **lwd**: spessore delle linee di livello.
  - **xlab, ylab**: etichette per gli assi X e Y.



FUNZIONE CONTOUR()

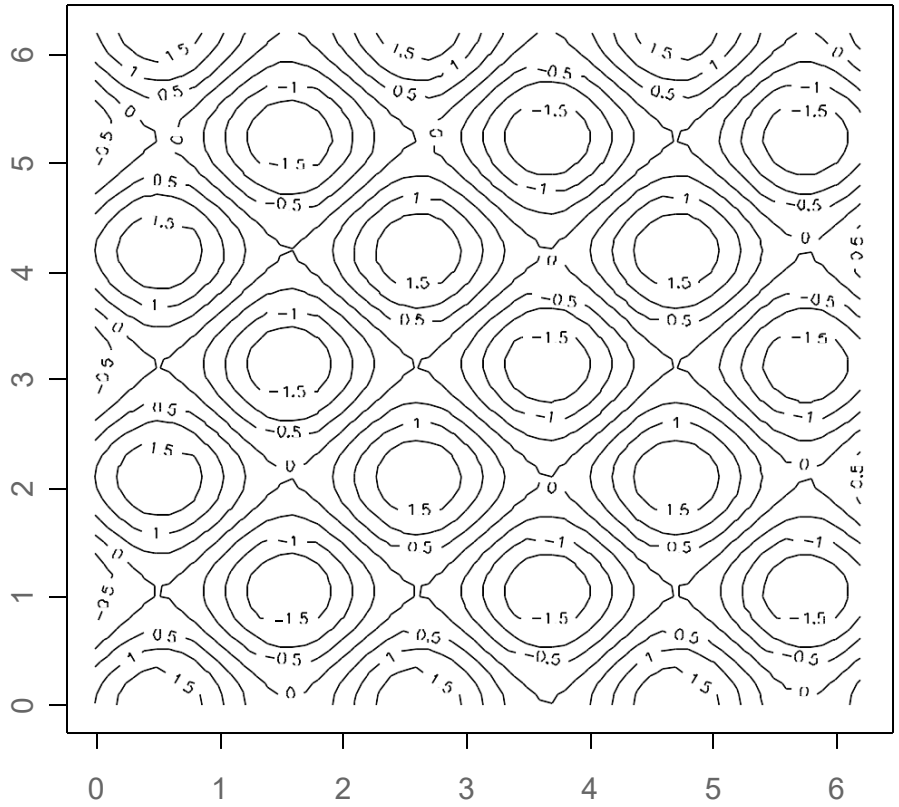
Rappresentazione della funzione: f(x,y) = sin(3x) + cos(3y)

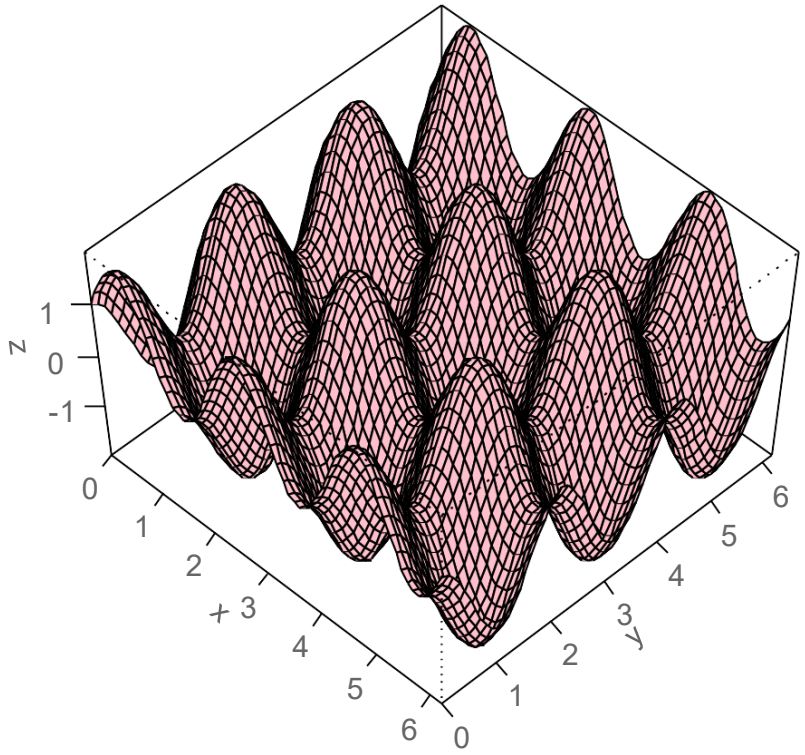



```
> x<-y<-seq(0,2*pi,by=0.1)
> z <- outer(sin(3*x),cos(3*y),"+")      crea la matrice z delle altezze
>  contour(x,y,z)
```

FUNZIONE CONTOUR()

Rappresentazione della funzione: f(x,y) = sin(3x) + cos(3y)

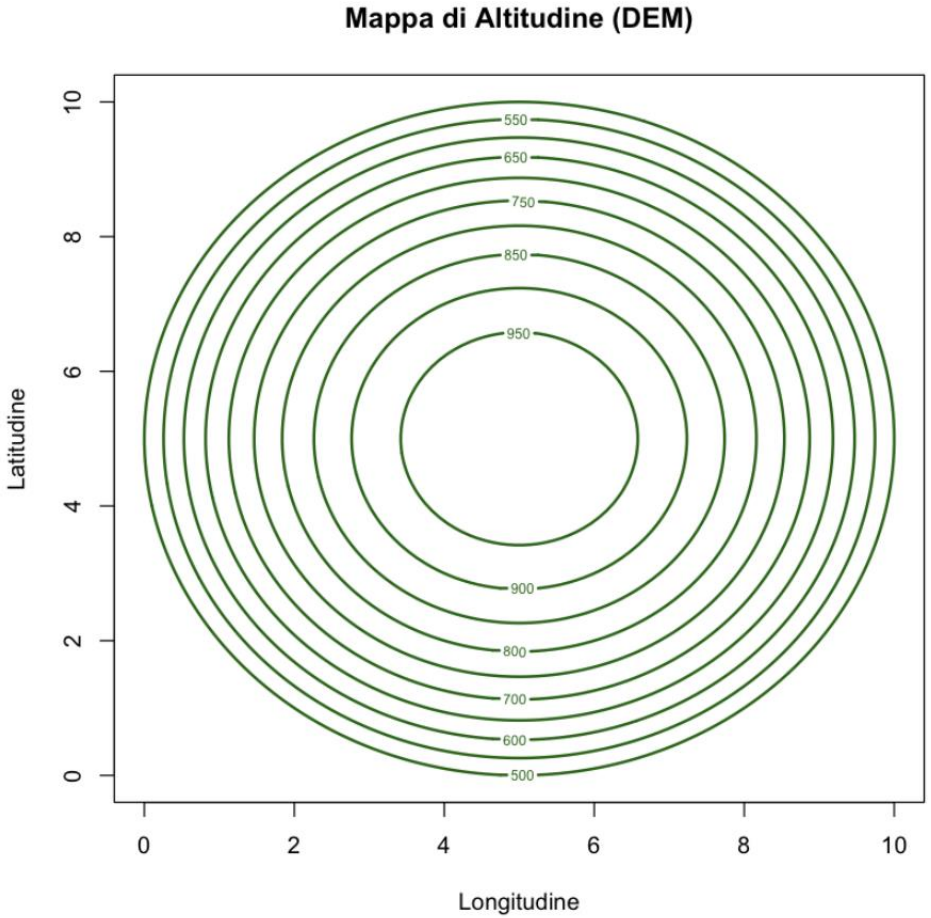





```
> x<-y<-seq(0,2*pi,by=0.1)
> z <- outer(sin(3*x),cos(3*y),"+")        # crea la matrice z delle altezze
>  contour(x,y,z)
```

FUNZIONE CONTOUR()

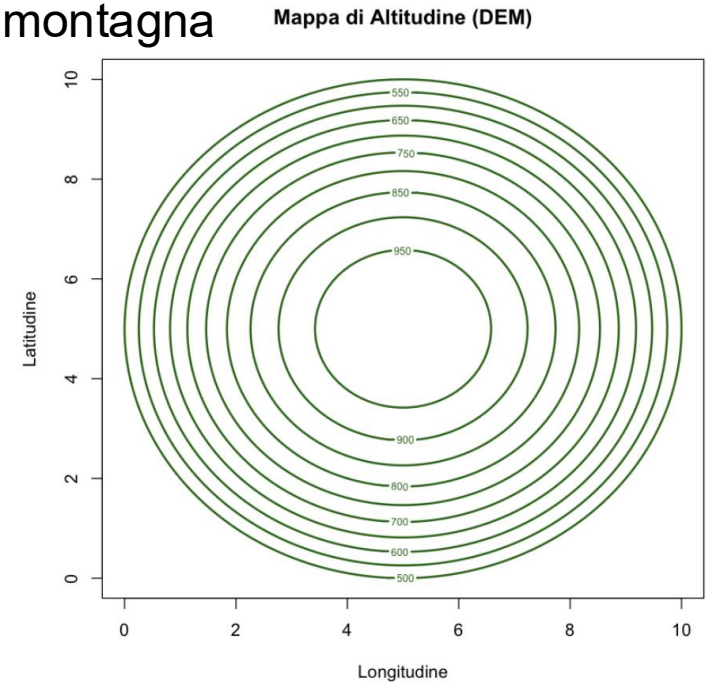
- Un esempio reale di applicazione della funzione contour() in un sistema di GIS (Geographic Information System) è la creazione di mappe di altitudine o mappe topografiche
- Nei sistemi GIS, i dati geografici come l'elevazione di una zona sono spesso rappresentati tramite curve di livello (contour lines) per mostrare aree con la stessa altitudine



FUNZIONE CONTOUR()

- Esempio: Generiamo una mappa di altitudine utilizzando curve di livello di una montagna

```r
# Creiamo un esempio di dati di elevazione

# Definire i punti sulla griglia
x <- seq(0, 10, length.out = 100)  # Coordinata longitudinale
y <- seq(0, 10, length.out = 100)  # Coordinata latitudinale

# Creiamo una matrice di elevazione, simulando una montagna
z <- outer(x, y, function(x, y) 1000 - ((x - 5)^2 + (y - 5)^2) * 20)

# Generiamo il grafico delle curve di livello
contour(x, y, z, levels = seq(500, 1000, by = 50), col = "darkgreen", lwd = 2,
        xlab = "Longitudine", ylab = "Latitudine", main = "Mappa di Altitudine (DEM)")
```

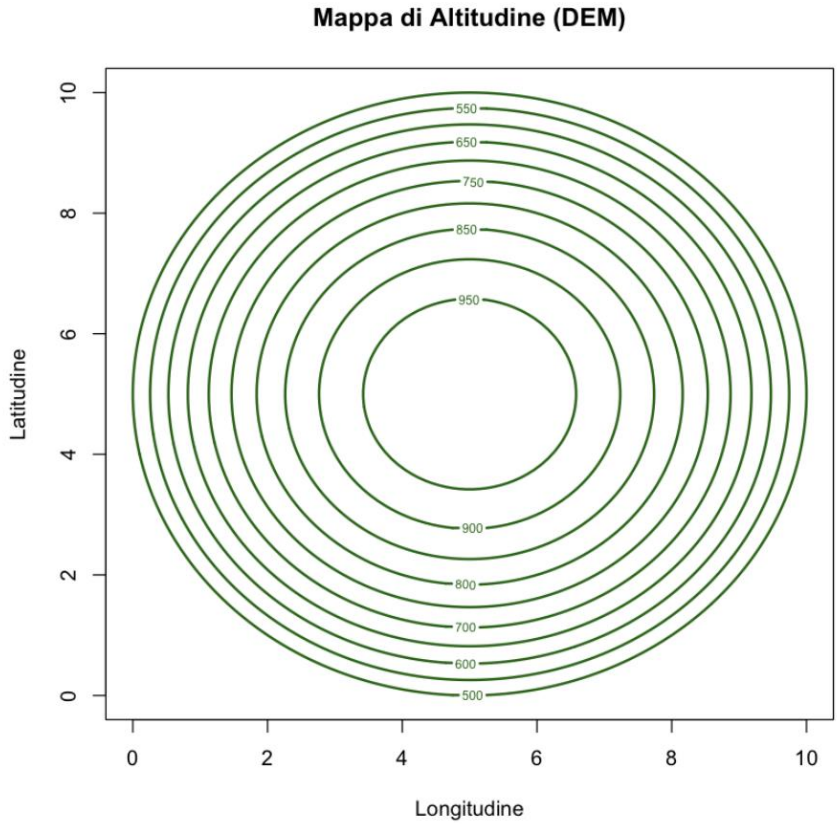


- **Definizione delle coordinate:**
  - seq(0, 10, length.out = 100): Crea una sequenza di 100 valori uniformemente distribuiti tra 0 e 10 sia per le coordinate x che per y.
- **Creazione della matrice di elevazione:**
  - outer(x, y, ...): Utilizza la funzione outer() per calcolare i valori di elevazione in base alla funzione specificata. Qui, il valore di elevazione diminuisce allontanandosi dal centro (5,5), simulando una montagna.

FUNZIONE CONTOUR()

- Generiamo una mappa di altitudine utilizzando curve di livello di una montagna

```r
# Creiamo un esempio di dati di elevazione

# Definire i punti sulla griglia
x <- seq(0, 10, length.out = 100)  # Coordinata longitudinale
y <- seq(0, 10, length.out = 100)  # Coordinata latitudinale

# Creiamo una matrice di elevazione, simulando una montagna
z <- outer(x, y, function(x, y) 1000 - ((x - 5)^2 + (y - 5)^2) * 20)

# Generiamo il grafico delle curve di livello
contour(x, y, z, levels = seq(500, 1000, by = 50), col = "darkgreen", lwd = 2,
        xlab = "Longitudine", ylab = "Latitudine", main = "Mappa di Altitudine (DEM)")
```



- **Generazione del grafico delle curve di livello:**
  - contour(...): Disegna le curve di livello utilizzando i valori di elevazione. Il parametro levels specifica i livelli di elevazione da visualizzare (da 500 a 1000 in incrementi di 50).

DOMANDE?