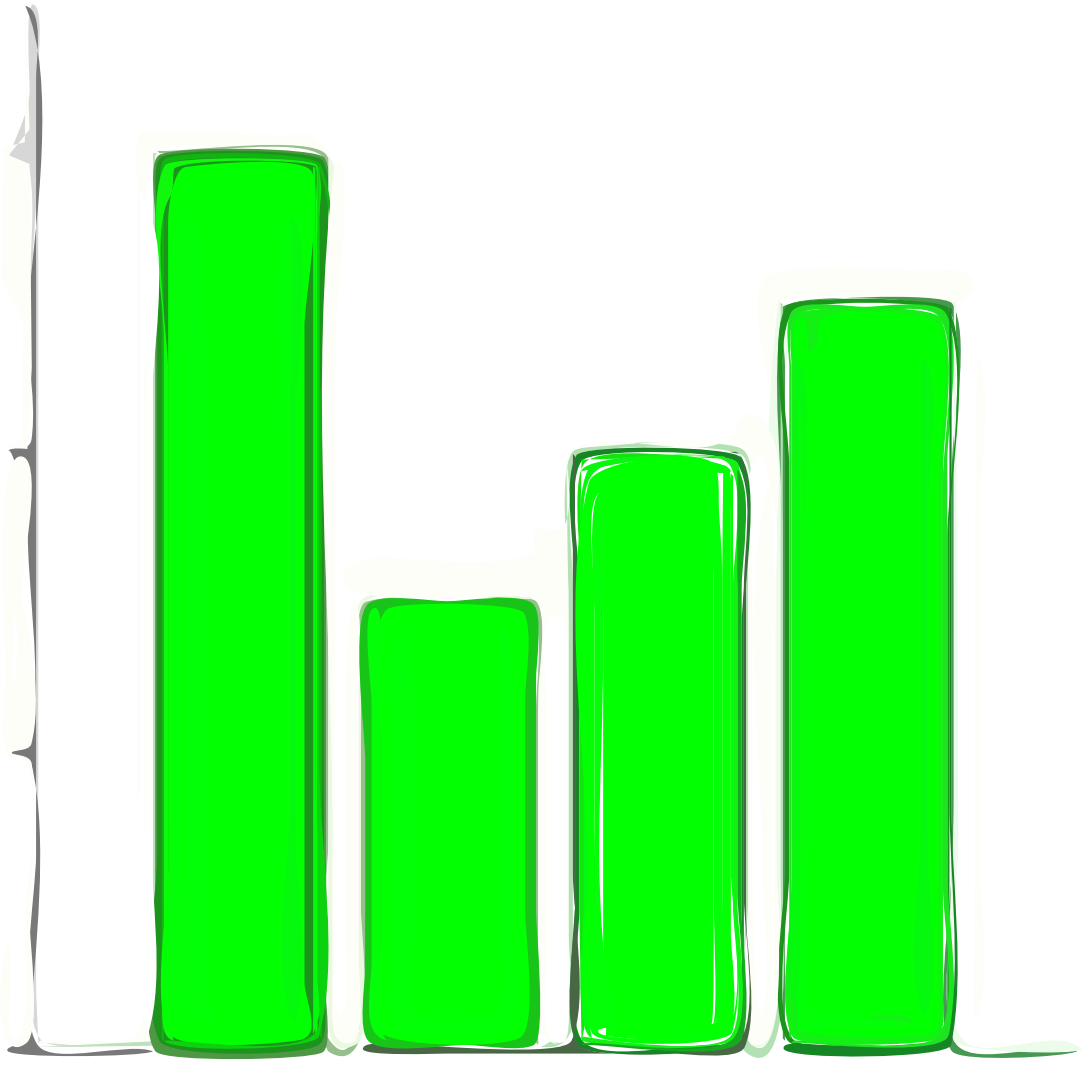
Grafici e diagrammi con Jgraph

Articolo estratto dall'opera «Informatica per sopravvivere»

Massimo Piai ⟨*pxam67 (@) virgilio·it*⟩

2006.02.18



Massimo Piai è un matematico appassionato di informatica, che ha trovato nel software libero e nella libertà delle informazioni l'unica possibilità di sviluppare tale passione. I suoi campi di interesse attuali sono la matematica e le scienze, la diffusione della Cultura Informatica, la didattica e la pedagogia.

Il presente lavoro è stato realizzato utilizzando Alml, il sistema di composizione SGML realizzato da Daniele Giacomini per la gestione dei suoi *Appunti di informatica libera*.

Indice generale

Introduzione

Jgraph[1] è un programma che serve a tracciare grafici e diagrammi strutturati. Diversamente da altri strumenti per il disegno, Jgraph è un programma **non interattivo**: legge un file di input (il ***sorgente***) e produce un file di output (la ***presentazione***) in formato PostScript oppure EPS. L'output può in seguito essere quindi visualizzato, stampato, incorporato in un altro file per ulteriori elaborazioni oppure convertito in altri formati.

Inoltre, si noti che Alml (il sistema di composizione SGML di Daniele Giacomini) offre supporto - a partire da gennaio 2006 - per il linguaggio Jgraph.

Questo capitolo costituisce una introduzione di livello elementare all'uso e alle caratteristiche principali del programma, senza alcuna prestesa di essere esauriente; la documentazione di riferimento più completa per Jgraph è costituita dalla pagina di manuale, cui si rinvia il lettore interessato per gli eventuali approfondimenti.

---

[1] **Jgraph** GNU GPL

Invocazione

Ecco l’utilizzo tipico di Jgraph:

$ jgraph in.jgr > out.eps [*Invio*]

oppure:

$ jgraph -P in.jgr > out.ps [*Invio*]

o, ancora:

$ cat in.jgr | jgraph -P | ps2eps -f -l -q > out.eps [*Invio*]

e simili.

In assenza di un nome di file dato come argomento, Jgraph legge dallo standard input.

Esempi elementari

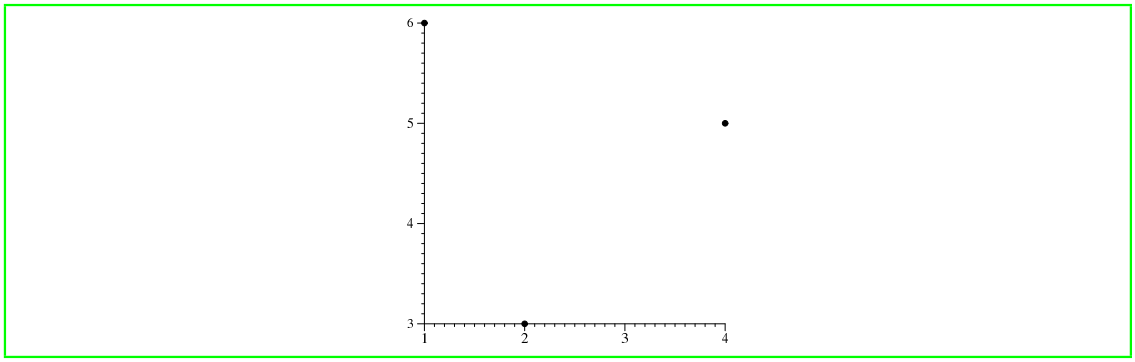
Al fine di prendere confidenza con il programma Jgraph, cominciamo vedere alcuni semplici esempi. Per indicare a Jgraph di tracciare un nuovo grafico o diagramma si utilizza la parola chiave '**`newgraph`**'; quindi si aggiungono delle curve al diagramma mediante '**`newcurve`**'; infine si aggiungono dei punti alla curva mediante '**`pts`**'.

Il listato 2.1 mostra un esempio elementare, presentato graficamente in figura 2.2. Si tratta semplicemente di disegnare tre punti: (2,3), (4,5) e (1,6). Jgraph si occupa di aggiungere gli assi di riferimento e di scegliere lo stile di tracciamento dei punti; è un esempio decisamente banale, tanto per iniziare.

Listato 2.1. Jgraph: un esempio elementare.

```
(* Simple jgraph *)

newgraph

newcurve pts 2 3   4 5   1 6
```

Figura 2.2.



Si considerino adesso il listato 2.3 e la corrispondente figura 2.4. Si tratta di un esempio leggermente più complesso: nelle righe 1-3 si dà inizio al grafico mediante '**`newgraph`**', quindi si definiscono le dimensioni degli assi;[1] le righe 5-9 tracciano tre curve: nella prima si lascia che sia Jgraph a scegliere lo stile di tracciamento, nella seconda si specifica di indicare i punti mediante triangoli connessi da una linea solida, nella terza si specifica di non indicare i punti ma di tracciare solamente la linea che li unisce (la linea sarà tratteggiata e di colore rosso[2]). Si noti che l'intervallo visibile degli assi è stato automaticamente calcolato in modo da contenere precisamente tutti i punti specificati nel sorgente.

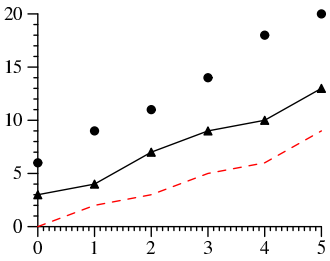
Nella pagina di manuale si trovano definiti i possibili stili da associare allle parole chiave '**`marktype`**' e '**`linetype`**'. Si noti in particolare che '**`newline`**' è semplicemente un sinonimo per '**`newcurve marktype none linetype solid`**'.

Listato 2.3. Jgraph: un grafico con tre curve di tipo differente.

```
1  newgraph
2  xaxis size 2
3  yaxis size 1.5
4
5  newcurve pts 0 6  1 9  2 11  3 14  4 18 5 20
6  newcurve marktype triangle linetype solid
7           pts 0 3  1 4  2 7  3 9  4 10 5 13
8  newcurve marktype none linetype dashed color 1 0 0
```

```
9           pts 0 0  1 2  2 3  3 5  4 6  5 9
```

Figura 2.4.



[1] in pollici

[2] I colori vanno specificati in forma codificata mediante terne RGB, con valore uno che indica la massima saturazione.

Alcuni dettagli sulla sintassi

L'algoritmo di riconoscimento dell'input da parte di Jgraph è basato non sulle righe bensì sugli elementi sintattici (*token-based*); Jgraph semplicemente lavora su parole separate da spazi bianchi; i commenti devono essere inclusi fra '`(*`' e '`*)`'. Gli oggetti fondamentali per Jgraph sono:

- la ***pagina***;
- i ***grafici***;
- gli ***assi***;
- le ***curve***;
- le ***stringhe***;
- le ***legende***.

Si può pensare che, quando si opera sul sorgente, si stanno effettivamente creando o modificando tali oggetti nella presentazione. Pertanto, quando si scrive '`newcurve`', si sta in effetti modificando o creando una curva, per la quale è possibile specificare ***attributi***, come '`marktype`' e '`linetype`', e dei punti. La maggior parte degli attributi possiedono un valore predefinito che però è possibile cambiare. Se si specifica un attributo più di una volta Jgraph considera valida l'ultima indicazione, sicché ad esempio '`newcurve marktype box marktype circle`' equivale a '`newcurve marktype circle`'.

In parziale deroga a quanto or ora specificato, la parola chiave '`pts`' agisce in maniera «additiva», pertanto '`newcurve pts 0 0 1 1 2 2`' equivale a '`newcurve pts 0 0 pts 1 1 pts 2 2`'.

È inoltre possibile includere dei file esterni nel sorgente Jgraph mediante '`include` ***nome_file***'.

Gestione degli assi

Se capita di non gradire la maniera in cui Jgraph produce automaticamente gli assi, può tornare utile la tabella 4.1.

Tabella 4.1. Jgraph: alcuni attributi degli assi.

Attributo ed eventuali valori	Significato
`size` ***dimensione***	Imposta la dimensione dell'asse a ***dimensione*** (in pollici).
`min` ***valore***	Imposta il valore minimo a ***valore***.
`max` ***valore***	Imposta il valore massimo a ***valore***.
`hash` ***numero_valori***	Traccia una tacca (primaria) e un'etichetta di tacca ogni ***numero_valori*** valori.
`mhash` ***numero_tacche***	Traccia ***numero_tacche*** tacche secondarie fra ogni due tacche primarie consecutive.
`gray` ***scala_di_grigio***	Imposta la scala di grigio dell'asse a ***scala_di_grigio***, ove zero significa nero e uno significa bianco.
`color` ***componente_rossa*** ↩ ↪ ***componente_verde*** ↩ ↪ ***componente_azzurra***	Imposta il colore dell'asse, codificato mediante la terna RGB indicata.
`nodraw`	Impedisce il tracciamento dell'asse, in ogni sua parte.
`draw`	Garantisce il tracciamento dell'asse, in ogni sua parte.
`log`	Imposta la scala dell'asse come logaritmica.
`linear`	Imposta la scala dell'asse come lineare.
`no_draw_hash_marks`	Impedisce il tracciamento delle tacche lungo l'asse.
`no_draw_hash_labels`	Impedisce il tracciamento delle etichette lungo l'asse.
`draw_at` ***valore***	Traccia l'asse in una posizione diversa dal valore minimo.
`label` ***etichetta***	Imposta l'etichetta dell'asse.

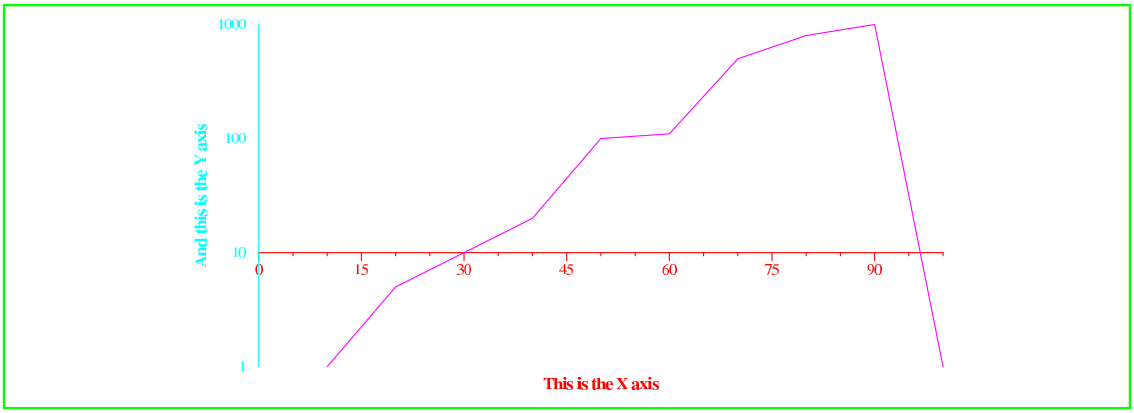
Il listato 4.2 e la figura 4.3 illustrano l'uso di alcuni degli attributi descritti nella tabella 4.1.

Listato 4.2. Jgraph: uso di alcuni degli attributi degli assi.

```
newgraph

xaxis
  size 6
  min 0 max 100
  hash 15 mhash 2 (* i.e. minor hashes at the 5's and 10's *)
  color 1 0 0
  label : This is the X axis
  draw_at 10

yaxis
```

```
   size 3
   min 1 max 1000
   log
   no_draw_hash_marks
   label : And this is the Y axis
   color 0 1 1

newline color 1 0 1
   pts 10 1
       20 5
       30 10
       40 20
       50 100
       60 110
       70 500
       80 800
       90 1000
       100 1
```

Figura 4.3.



Gestione delle curve

Jgraph assume dei valori predefiniti per le dimensioni degli ***indicatori di punto***, ma è possibile impostarli diversamente mediante '**`marksize`** ***larghezza altezza***'. La larghezza si intende nelle unità dell'***asse delle ascisse***, a meno che la scala non sia logaritmica nel qual caso si intende espressa in pollici; analogamente l'altezza fa riferimento all'unità dell'***asse delle ordinate***.

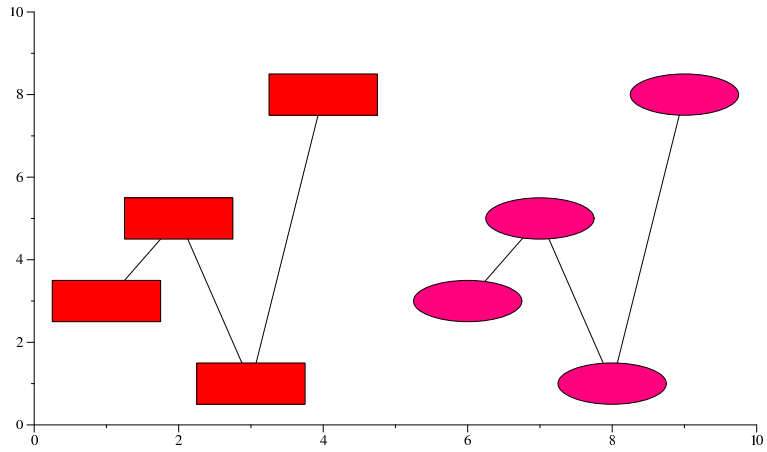
La parola chiave '**`copycurve`**' permette di creare una curva avente i medesimi attributi di quella precedente (ma non gli stessi punti, ovviamente).

Il listato 5.1 e la figura 5.2 illustrano un esempio. Si noti che '**`cfill`**' riempie l'interno degli indicatori con il colore specificato.

Listato 5.1. Jgraph: esempio di gestione delle curve.

```
newgraph

xaxis min 0 max 10 size 7
yaxis min 0 max 10 size 4

newcurve marktype box marksize 1.5 1 linetype solid cfill 1 0 0
   pts 1 3  2 5  3 1  4 8

copycurve marktype ellipse cfill 1 0 .5
   pts 6 3  7 5  8 1  9 8
```

Figura 5.2.



Stringhe

La gestione delle stringhe in Jgraph avviene mediante la parola chiave '**`newstring`**'. Gli attributi delle stringhe sono la ***posizione***, la ***fonte***, la ***dimensione della fonte***, il ***colore***, la ***rotazione***, l'***allineamento*** e il ***testo***. La posizione si imposta mediante '**`x`** ***posizione_x*** **`y`** ***posizione_y***'. La fonte dev'assere una fonte PostScript. Le fonti standard sono '**`Times-Roman`**', '**`Times-Italic`**', '**`Times-Bold`**', '**`Helvetica`**', '**`Helvetica-Italic`**', '**`Helvetica-Bold`**', '**`Courier`**' e '**`Symbol`**'. Il colore viene indicato tramite '**`lcolor`**', oppure si utilizza '**`lgray`**' per indicare la scala di grigio. Gli allineamenti si specificano come segue:

- '**`hjl`**', '**`hjr`**' e '**`hjc`**' corrispondono a sinistra, destra e centro (orizzontale) rispettivamente;
- '**`vjt`**', '**`vjb`**' e '**`vjc`**' corrispondono a sopra, sotto e centro (verticale) rispettivamente.

Si può ruotare una stringa mediante '**`rotate`** ***gradi***'. Si imposta il testo di una stringa mediante il carattere '**`:`**', seguito da uno spazio bianco e quindi dal testo vero e proprio. Per ottenere del testo su più righe è possibile utlizzare il carattere '**`\`**' al termine di ciascuna riga tranne l'ultima.

Anche le ***etichette*** dei grafici sono stringhe, quindi possono utilizzare gli attributi or ora descritti.

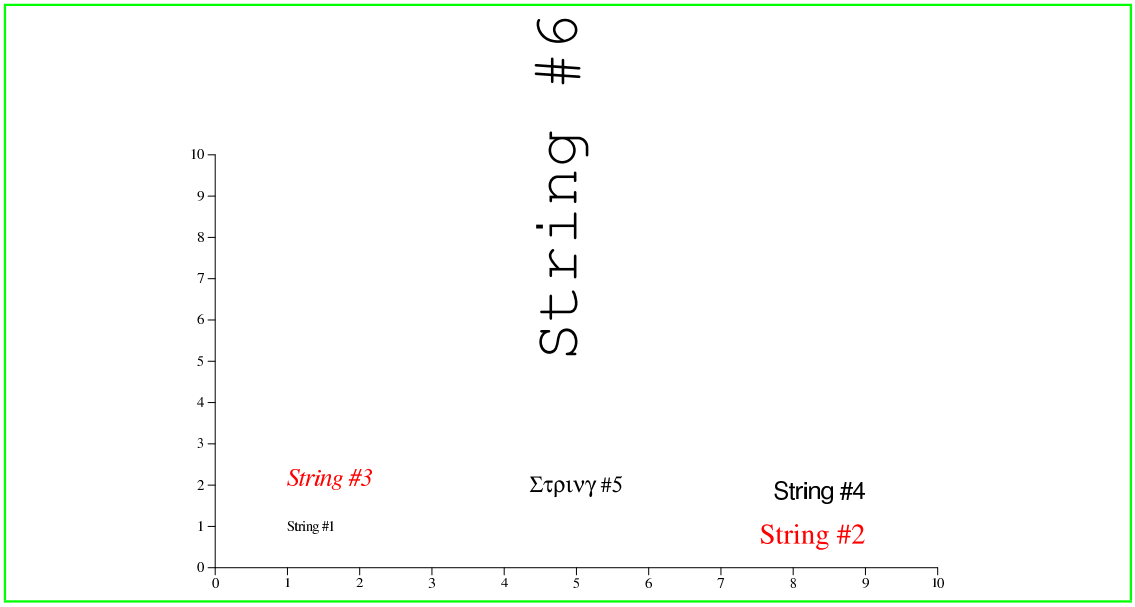
'**`copystring`**' copia tutti gli attributi di una stringa (testo incluso).

Il listato 6.1 e la figura 6.2 illustrano un esempio.

Listato 6.1. Jgraph: gestione delle stringhe.

```
newgraph
  xaxis min 0 max 10 hash 1 mhash 0 size 7
  yaxis min 0 max 10 hash 1 mhash 0 size 4

newstring hjl vjc x 1 y 1 : String #1

newstring hjr vjt x 9 y 1 fontsize 20 lcolor 1 0 0 : String #2

copystring hjl vjb x 1 y 2 fontsize 16 font Times-Italic : String #3

newstring hjr vjt x 9 y 2 fontsize 16 font Helvetica : String #4

newstring hjc vjc x 5 y 2 fontsize 16 font Symbol : String #5

newstring hjl vjb fontsize 45 font Courier rotate 90 x 5 y 5 : String #6
```

Figura 6.2.



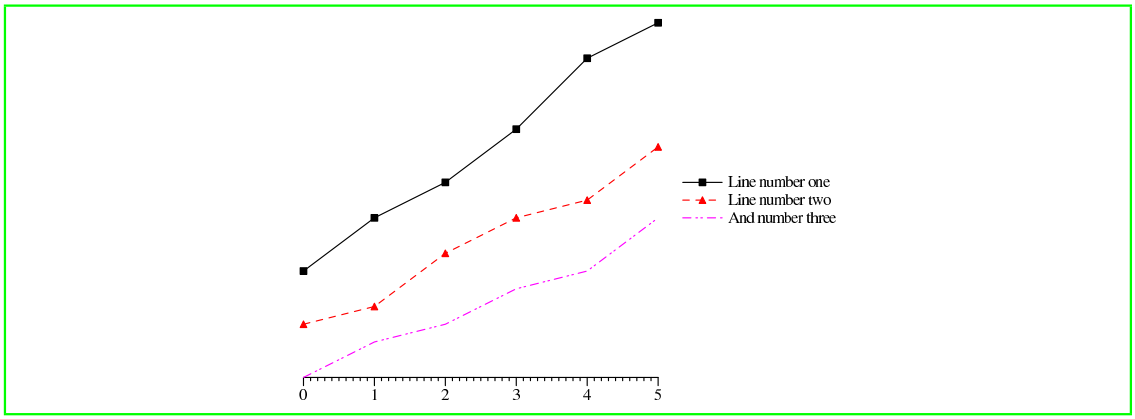
Legende

È possibile applicare un'etichetta a una curva mediante l'attributo '**`label`**': esso indica a Jgraph di creare una voce in ***legenda*** per la curva data. Il listato 7.1 e la figura 7.2 forniscono un esempio.

Listato 7.1. Jgraph: esempio di legenda.

```
newgraph
xaxis min 0 max 5
yaxis min 0 nodraw

newcurve marktype box linetype solid label : Line number one
   pts 0 6  1 9  2 11  3 14  4 18 5 20
newcurve marktype triangle linetype dashed color 1 0 0 label : Line number two
   pts 0 3  1 4  2 7  3 9  4 10 5 13
newcurve marktype none linetype dotdotdash color 1 0 1 label : And number three
         pts 0 0  1 2  2 3  3 5  4 6  5 9
```

Figura 7.2.

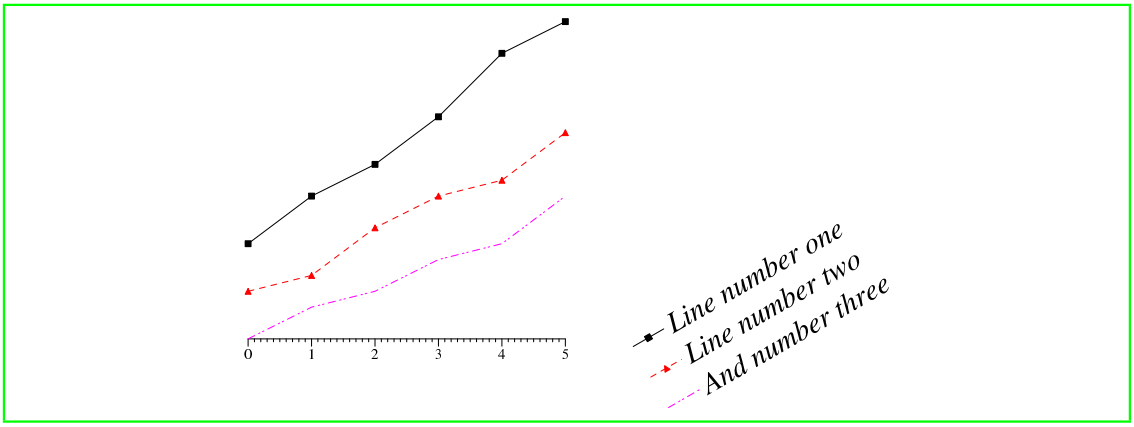


Si possono cambiare la fonte e la posizione della legenda, come unità complessiva, scrivendo '**`legend defaults`**' e specificando gli attributi per le stringhe: in tal modo si modificano tutte le stringhe della legenda. Il listato 7.3 e la figura 7.4 forniscono un esempio; è istruttivo provare a modificarlo e vedere quali effetti ne sortiscano. Le legende hanno altre interessanti caratteristiche, che si trovano ben descritte nella pagina di manuale.

Listato 7.3. Jgraph: un altro esempio di legenda.

```
newgraph
xaxis min 0 max 5
yaxis min 0 nodraw

newcurve marktype box linetype solid label : Line number one
   pts 0 6  1 9  2 11  3 14  4 18 5 20
newcurve marktype triangle linetype dashed color 1 0 0 label : Line number two
   pts 0 3  1 4  2 7  3 9  4 10 5 13
newcurve marktype none linetype dotdotdash color 1 0 1 label : And number three
         pts 0 0  1 2  2 3  3 5  4 6  5 9

legend defaults font Times-Italic fontsize 20 rotate 30 hjl vjt x 6 y 0
```

Figura 7.4.



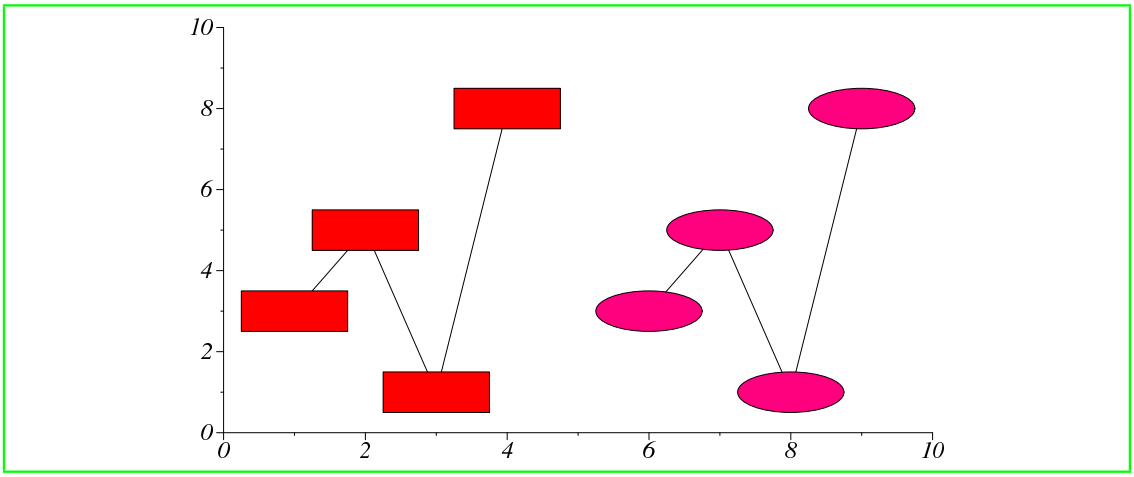
Etichette di tacca

Analogamente al caso delle legende è possibile modificare l'insieme delle ***etichette di tacca*** di un asse, come unità complessiva, mediante l'attributo '**`hash_labels`**': ad esempio, per cambiare le etichette di tacca in figura 5.2 in maniera tale che la fonte sia '**`Times-Italic`**' e la dimensione della fonte sia 16, si può scrivere '**`hash_labels font Times-Italic fontsize 16`**', e analogamente per l'asse delle ordinate (listato 8.1, righe 4 e 7, e figura 8.2).

Listato 8.1. Jgraph: gestione delle etichette di tacca.

```
1   newgraph
2
3   xaxis min 0 max 10 size 7
4     hash_labels font Times-Italic fontsize 16
5
6   yaxis min 0 max 10 size 4
7     hash_labels font Times-Italic fontsize 16
8
9   newcurve marktype box marksize 1.5 1 linetype solid cfill 1 0 0
10     pts 1 3  2 5  3 1  4 8
11
12  copycurve marktype ellipse cfill 1 0 .5
13     pts 6 3  7 5  8 1  9 8
```

Figura 8.2.



Grafici a barre

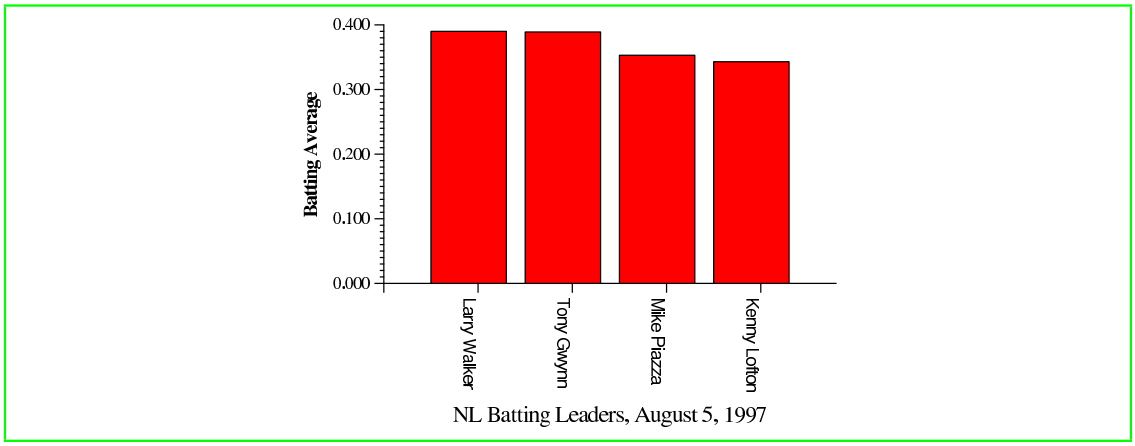
Utilizzando come indicatori ‘**xbar**’ e ‘**ybar**’ si ottengono i grafici a barre: ‘**xbar**’ produce barre che si estendono parallelamente all’asse delle ordinate, ‘**ybar**’ barre che si estendono parallelamente all’asse delle ascisse.

Il listato 9.1 e la figura 9.2 forniscono un interessante esempio di grafico a barre.

Listato 9.1. Jgraph: grafico a barre.

```
1   newgraph
2
3   xaxis
4     min 0.1 max 4.9
5     size 3.5
6     hash 1 mhash 0 no_auto_hash_labels
7
8   yaxis
9     min 0 max .4
10    size 2
11    precision 3
12
13  newcurve marktype xbar cfill 1 0 0 marksize .8
14    pts   1 .390
15          2 .389
16          3 .353
17          4 .343
18
19  xaxis
20    hash_label at 1 : Larry Walker
21    hash_label at 2 : Tony Gwynn
22    hash_label at 3 : Mike Piazza
23    hash_label at 4 : Kenny Lofton
24    hash_labels hjl vjc font Helvetica rotate -90
25
26  yaxis label : Batting Average
27  title : NL Batting Leaders, August 5, 1997
```

Figura 9.2.



Alcune osservazioni in merito al listato 9.1:

- **Righe 3-6**

  Si imposta la dimensione e l'intervallo dei valori per l'asse delle ascisse e si impostano le tacche sulle unità senza tacche secondarie; inoltre si indica a Jgraph di non generare le etichette di tacca in modo automatico, poiché si desidera personalizzarle.

- **Righe 8-11**

  Niente di particolare da notare tranne l'attributo '`precision`' il quale specifica di includere tre decimali nelle etichette di tacca per l'asse delle ordinate.

- **Righe 13-17**

  Traccia la curva tramite barre rosse verticali.

- **Righe 19-24**

  Ulteriore modifica che interessa l'asse delle ascisse, generando etichette di tacca per ciascun punto e successivamente modificandole in maniera che appaiano ruotate di 90 gradi (in senso orario).

- **Righe 26-27**

  Si genera un etichetta per l'asse delle ordinate e un titolo per il grafico.

L'esempio illustrato nella presente sezione dovrebbe convincere delle possibilità del programma Jgraph: è possibile generare grafici anche di una certa complessità in maniera abbastanza naturale.

Particolarità

10.1 Poligoni

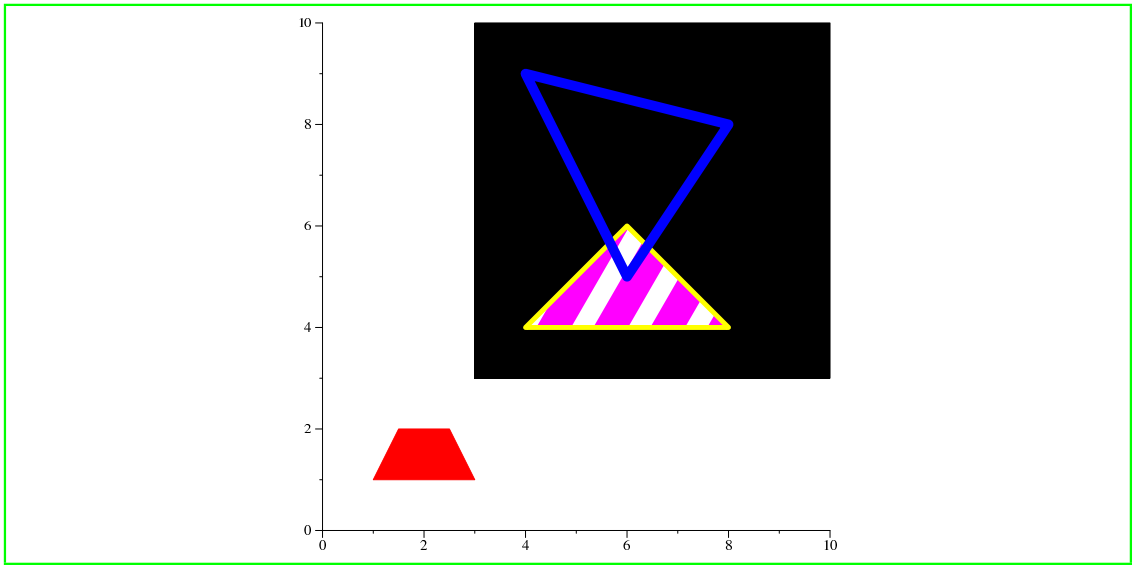
La possibilità di tracciare dei ***poligoni*** estende non poco le capacità grafiche del programma Jgraph: è sufficiente applicare l'attributo '**`poly`**' alla curva da tracciare. Gli attributi '**`pfill`**' e '**`pcfill`**' permettono inoltre di specificare una scala di grigio o un colore per la campitura del poligono. L'attributo '**`linethickness`**' consente di controllare lo spessore del perimetro.[1] È possibile campire il poligono con un disegno a strisce in alternativa alla campitura solida mediante '**`ppattern stripe`** ***inclinazione***', ove ***inclinazione*** controlla l'inclinazione delle strisce. Infine, se si indica `-1` come colore di campitura del poligono viene tracciato solamente il bordo mentre l'interno rimane vuoto, il che può essere talvolta desiderabile.

Il listato 10.1 e la figura 10.1 presentano un esempio con alcuni poligoni di diversa campitura. Si osservi che Jgraph traccia le curve nell'ordine indicato, sicché è possibile prevedere le eventuali sovrapposizioni.

Listato 10.1. Jgraph: poligoni.

```
newgraph

xaxis min 0 max 10 size 5
yaxis min 0 max 10 size 5

(* Draw a red trapezoid -- setting both the fill and color to be red means
   that the entire trapezoid, and not just the interior, will be red. *)

newline poly pcfill 1 0 0 color 1 0 0
   pts 1 1  3 1   2.5 2  1.5 2

(* Draw a big black square *)

newline poly pfill 0 pts 3 3  10 3  10 10  3 10

(* Draw a thick yellow triangle with a purple, striped interior inside the black
   square *)

newline poly linethickness 5 color 1 1 0 pcfill 1 0 1 ppattern stripe 60
   pts  4 4   8 4  6 6

(* Draw a blue triangle with a thick border no fill *)

newline poly linethickness 10 color 0 0 1 pfill -1 pts 4 9  6 5 8 8
```

Figura 10.2.



10.2 Curve di Bézier

Si possono generare delle linee curve con Jgraph se si interpretano alcuni dei punti indicati come ***punti di controllo*** per una ***curva di Bézier***.

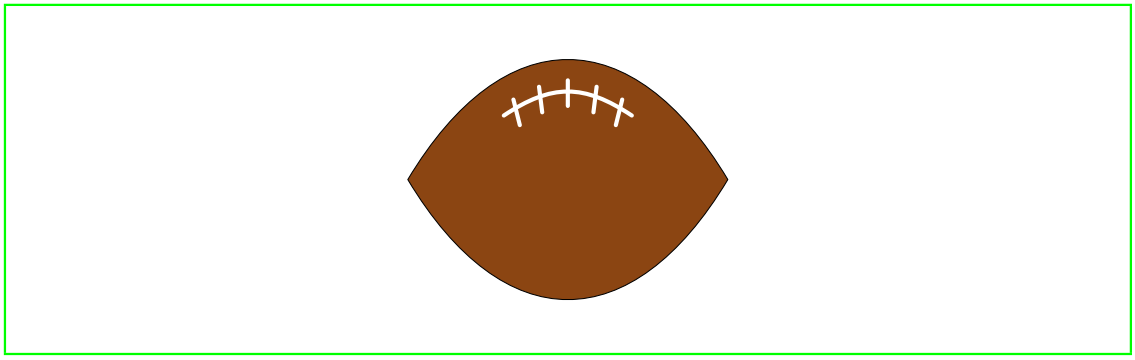
Nella sezione 13 vengono indicati dei possibili spunti di approfondimento sulla questione, ma si consideri che per definire una curva di Bézier con Jgraph servono almeno quattro punti, e in generale è possibile tracciare una curva di Bézier mediante 3****n***+1 punti di controllo, di cui 2****n*** di controllo e i rimanenti ***n***+1 di passaggio della curva.[2]

Il listato 10.3 e la figura 10.4 fornuscono un esempio in cui viene disegnato un pallone da football americano (molto stilizzato a dire il vero) mediante curve di Bézier.

Listato 10.3. Jgraph: curve di Bézier.

```
newgraph
xaxis min 0 max 10 nodraw
yaxis min 0 max 10 nodraw

(* Draw the outline of the football *)
newline bezier poly pcfill .543 .270 .074 pts
  0 5   3 10   7 10    10 5
        7 0    3 0     0 5

(* Draw the main thread *)

newline bezier linethickness 4 gray 1 pts
  3 7 4.5 8 5.5 8 7 7

(* Draw the crossthreads *)

copycurve nobezier pts 3.5 6.7 3.3 7.5
copycurve pts 6.5 6.7 6.7 7.5
copycurve pts 4.2 7.1 4.1 7.9
copycurve pts 5.8 7.1 5.9 7.9
copycurve pts 5 7.3  5 8.1
```

Figura 10.4.



[1] in unità assolute PostScript, valore predefinito 1,0

[2] con ***n*** intero positivo

Sezione 11

Altre caratteristiche

Accenniamo ad alcune ulteriori interessanti caratteristiche del programma Jgraph, rinviando il lettore interessato alla sezione 13 per eventuali approfondimenti.

Dall'interno di un sorgente Jgraph è possibile invocare un comando della shell o un programma esterno scrivendo '**`shell :`** ***comando***'; l'output generato viene contestualmente incluso nel sorgente.

Ovviamente si tratta di una caratteristica estremamente interessante e potente, poiché permette di utilizzare strumenti come Perl, Awk e simili per il tracciamento di grafici di funzione, estrazione e manipolazione di dati, ecc.; parallelamente si tratta di una caratteristica **potenzialmente pericolosa**, in quanto Jgraph non effettua nessun controllo sul comando che viene eseguito, aprendo così il varco a possibili cavalli di Troia.
Prestare dunque un'adeguata cautela!

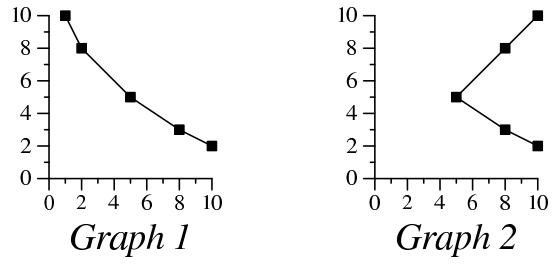
Un'altra caratteristica interessante di Jgraph è quella di poter utilizzare una qualunque immagine in formato EPS come indicatore, utilizzando l'attributo di curva '**`eps`**'.

Infine, si noti che è possibile tracciare più di un grafico nella stessa immagine mediante un ulteriore '**`newgraph`**' oppure mediante '**`copygraph`**': in quest'ultimo caso il grafico successivo eredita gli assi dal precedente. Il posizionamento relativo dei vari grafici è gestibile mediante gli attributi di grafico '**`x_translate`**' e '**`y_translate`**'. È altresì possibile posizionare i vari grafici su più pagine, mediante la parola chiave '**`newpage`**'.[1] Il listato 11.1 e la figura 11.2 presentano un esempio di utilizzo di '**`copygraph`**'.

Listato 11.1. Jgraph: grafici multipli.

```
newgraph
xaxis min 0 max 10 size 1 label fontsize 16 font Times-Italic : Graph 1
yaxis min 0 max 10 size 1

newcurve marktype box linetype solid pts 1 10 2 8 5 5 8 3 10 2

copygraph
x_translate 2
xaxis label : Graph 2
newcurve marktype box linetype solid pts 10 10 8 8 5 5 8 3 10 2
```

Figura 11.2.



[1] Ovviamente si tratta di una possibilità compatibile solo con l’output in formato PostScript (non EPS).

Utilizzo programmatico

Il punto di forza dei programmi non interattivi è quello di poter essere facilmente utilizzati come anelli di una catena di strumenti o processi; la connessione delle varie parti può essere gestita semplicemente mediante generiche pipeline, o addirittura mediante specifici programmi «wrapper», progettati per uno scopo preciso, che costruiscano programmaticamente l'input da far elaborare.

12.1 «njgraph.pl»

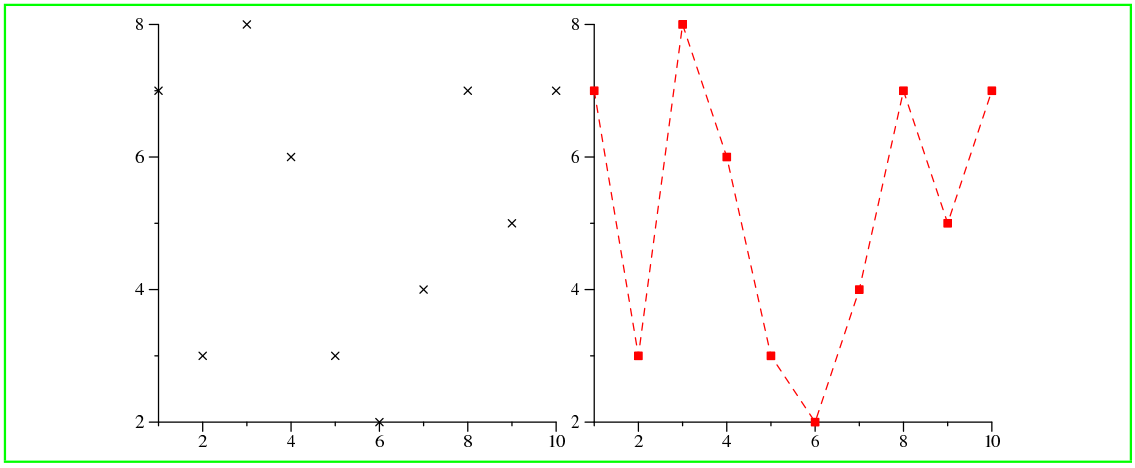
Un esempio interessante è il programma '**`njgraph.pl`**' di Neil Spring. Si tratta di un piccolo sorgente in Perl che in pratica realizza una specie di programma frontale per Jgraph: esso legge dallo standard input una singola serie di dati da rappresentare graficamente, e dopo averli opportunamente passati a Jgraph per l'elaborazione invoca GV, a meno che lo standard output non venga rediretto. Il programma '**`njgraph.pl`**' riconosce altresì gli attributi '**`marktype`**', '**`linetype`**' e '**`color`**' (e relativi valori) facoltativamente presenti sulla riga di comando, e li posiziona opportunamente all'interno del sorgente Jgraph prima dell'elaborazione.

La figura 12.1 illustra qualche esempio di invocazione del programma '**`njgraph.pl`**', mentre la figura 12.2 ne illustra l'output.

Figura 12.1. Invocazione del programma '**`njgraph.pl`**'.

```
$ njgraph.pl [ Invio ]

  automatically invoking gv; redirect if you want something else

1 7 2 3 3 8 4 6 5 3 6 2 7 4 8 7 9 5 10 7 [ Invio ]
[ Ctrl d ]
[ q ]
$ njgraph.pl marktype box linetype dashed color 1 0 0 [ Invio ]

  automatically invoking gv; redirect if you want something else

1 7 2 3 3 8 4 6 5 3 6 2 7 4 8 7 9 5 10 7 [ Invio ]
[ Ctrl d ]
[ q ]
$
```

Figura 12.2. Output del programma '**njgraph.pl**'.



12.2 «data2multijgr.sh». Qualche nota autobiografica

Pochi giorni dopo la pubblicazione del presente articolo, ho ricevuto un messaggio di posta elettronica da parte di un lettore. Ecco il frammento rilevante:

```
...

Per quanto riguarda la creazione dei grafici devo creare un grafico con
circa 200 barre orizzontali e pertanto devo creare un grafico che si
sviluppi su più pagine mettendo un certo numero di barre per ogni
pagina.

Distinti saluti

...
```

Ad ulteriore conferma della versatilità degli strumenti come Jgraph, ho realizzato in poco più di mezz'ora[1] il programma '**data2multijgr.sh**' (il file 'data2multijgr_sh' contenente il programma dovrebbe essere disponibile in allegato alla versione HTML del presente lavoro). Il programma, pur essendo molto rudimentale, risolve il problema del lettore. Eccone la sintassi:

```
data2multijgr.sh [opzione] …
```

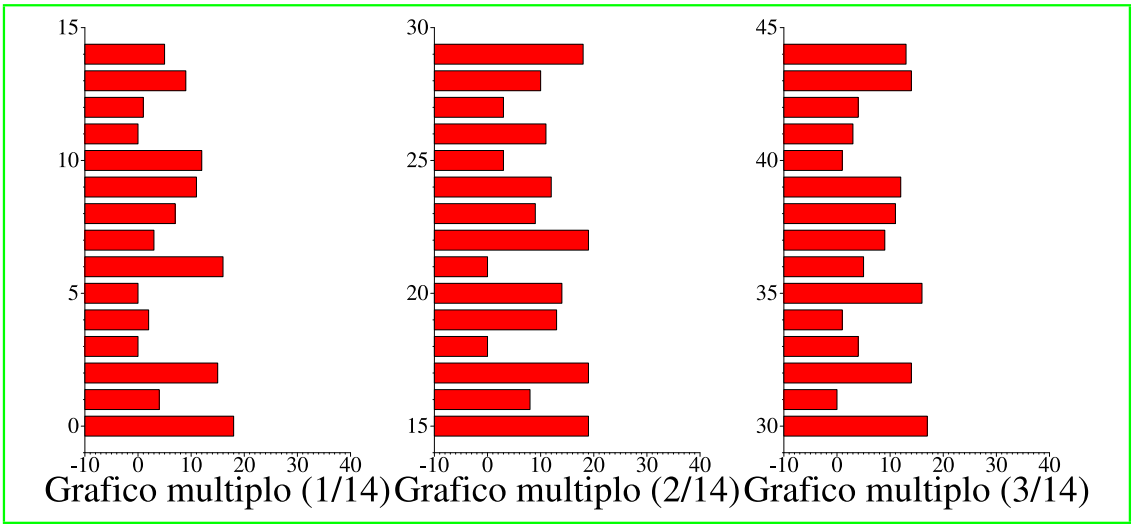
ed eccone un utilizzo tipico:

```
$ cat data [ Invio ]

18 4 15 0 2 0 16 3 7 11 12 0 1 9 5 19 8 19 0 13 14 0 19 9 12 3 11 3 10 18
17 0 14 4 1 16 5 9 11 12 1 3 4 14 13 1 13 13 1 5 6 7 18 17 16 10 13 19 13
15 17 3 15 3 7 8 19 4 18 3 16 11 6 13 5 11 14 19 4 15 16 2 2 14 12 10 5 17
9 10 12 18 5 19 13 13 8 13 17 18 8 6 9 14 19 14 6 13 5 2 1 2 17 15 8 1 18
13 18 19 4 2 18 9 13 3 14 1 8 4 11 16 2 0 11 13 7 9 18 4 11 19 18 0 7 7 1 5
12 11 16 8 5 6 18 19 10 4 12 18 0 4 15 2 16 6 15 15 15 6 0 18 5 10 11 4 17
4 1 2 16 18 10 1 4 0 0 14 5 5 13 17 9 8 0 17 6 7 13 13
```

```
$ cat data | data2multijgr.sh -d 15 -t "Grafico multiplo" -X 40 -x -10 >
multi.jgr [ Invio ]

$ cat multi.jgr [ Invio ]

  newgraph
  title fontsize 50 : Grafico multiplo (1/14)
  xaxis min -10 max 40 size 5 hash_labels fontsize 30
  yaxis min -1 max 15 size 8 hash_labels fontsize 30
  newcurve marktype ybar cfill 1 0 0 marksize 1 0.75 pts
  18 0 4 1 15 2 0 3 2 4 0 5 16 6 3 7 7 8 11 9 12 10 0 11 1 12 9 13 5 14
  newpage
  newgraph
  title fontsize 50 : Grafico multiplo (2/14)
  xaxis min -10 max 40 size 5 hash_labels fontsize 30
  yaxis min 14 max 30 size 8 hash_labels fontsize 30
  newcurve marktype ybar cfill 1 0 0 marksize 1 0.75 pts
  19 15 8 16 19 17 0 18 13 19 14 20 0 21 19 22 9 23 12 24 3 25 11 26 3 27 10 28 18 29
  newpage
  newgraph
  title fontsize 50 : Grafico multiplo (3/14)
  xaxis min -10 max 40 size 5 hash_labels fontsize 30
  yaxis min 29 max 45 size 8 hash_labels fontsize 30
  newcurve marktype ybar cfill 1 0 0 marksize 1 0.75 pts
  17 30 0 31 14 32 4 33 1 34 16 35 5 36 9 37 11 38 12 39 1 40 3 41 4 42 14 43 13 44
  newpage
  newgraph

  ...


$ cat multi.jgr | jgraph -P > multi.ps [ Invio ]

$
```

Il file ‘`multi.ps`’ che così si ottiene è un file PostScript composto da più pagine ciascuna contenente un grafico relativo a un segmento specifico dei dati (la figura 12.6 riporta per brevità solamente le prime tre pagine).

Figura 12.6. Jgraph: le prime tre pagine di una grafico multiplo, distribuito su 14 pagine, realizzato grazie al programma '**`data2multijgr.sh`**'.



La tabella 12.7 riassume le opzioni del programma. Chiunque fosse interessato è invitato adattare il programma '**`data2multijgr.sh`**' alle proprie esigenze, o aggiugervi delle caratteristiche più sofisticate (magari inviando una copia del programma modificato al sottoscritto).

Tabella 12.7. Opzioni del programma '**`data2multijgr.sh`**'.

Opzione	Descrizione
{`--dataperpage` | `-d`} ***dati_per_ogni_pagina***	Stabilisce quanti dati vengono inseriti nel grafico per ciascuna pagina.
{`--title` | `-t`} ***titolo***	Stabilisce il titolo da dare al grafico.
{`--xmin` | `-x`} ***valore_minimo***	Stabilisce il valore minimo sull'asse delle ascisse.
{`--xmax` | `-X`} ***valore_massimo***	Stabilisce il valore massimo sull'asse delle ascisse.
{`--ymin` | `-y`} ***valore_minimo***	Stabilisce il valore minimo sull'asse delle ordinate.
{`--ymax` | `-Y`} ***valore_massimo***	Stabilisce il valore massimo sull'asse delle ordinate.
{`--cfcolor` | `-c`} ***componente_rossa*** ↩ ↪ ***componente_verde*** ↩ ↪ ***componente_azzurra***	Imposta il colore delle barre, codificato mediante la terna RGB indicata.
{`--landscape` | `-l`}	Fa in modo che il risultato sia orientato in modo orizzontale. Il modo predefinito è quello verticale.
{`--psout` | `-p`}	Produce direttamente un file PostScript (attraverso Jgraph).
{`--view` | `-v`}	Visualizza direttamente il grafico (atraverso GV).

[1] A dire il vero il programma è stato successivamente limato, ma il prototipo funzionante era pronto dopo il tempo indicato (NdA).

Riferimenti e approfondimenti

- James S. Plank, *Jgraph -- A Filter for Plotting Graphs in Postscript*
  ⟨*http://www.cs.utk.edu/~plank/plank/jgraph/jgraph.html* ⟩
- Department of Computer Science, School of Engineering, University of Virginia
  - *Jgraph*
    ⟨*http://www.cs.virginia.edu/helpnet/Authoring_Tools/jgraph.html* ⟩
  - *Jgraph manpage*
    ⟨*http://www.cs.virginia.edu/cgi-bin/manpage?section=all&topic=jgraph* ⟩
- Justin Zobel, *Example jgraph scripts*
  ⟨*http://goanna.cs.rmit.edu.au/~jz/resources/jgraph.html* ⟩
- Ajay Shah, *A collection of examples using Jgraph*
  ⟨*http://www.mayin.org/ajayshah/KB/JGRAPH/index.html* ⟩
- *Wikipedia: Bézier curve*
  ⟨*http://en.wikipedia.org/wiki/B%C3%A9zier_curve* ⟩
- Neil Spring, *Neil's Software*
  ⟨*http://www.cs.umd.edu/~nspring/software/njgraph.pl* ⟩

Appendici

Appendice A

Informazioni aggiuntive sul software e altre opere citate

Jgraph, IV

GNU GPL

Massimo Piai *(... circa 1975)*
⟨*pxam67 (ad) virgilio·it* ⟩