
Subject: Re: Formatting plots in IDL

Posted by [sterne](#) on Thu, 27 Oct 1994 22:47:22 GMT

[View Forum Message](#) <> [Reply to Message](#)

>>>> > "David" == David Lynch <lynch@gst.gsfc.nasa.gov> writes:

David> I would like to make plots of correlated data in such a way that
David> two or more plots share the same x-axis, but are stacked
David> vertically with only a line separating them. I can use the
David> multi command to get several plots on one page, but these are
David> separated by a lot of space and x-axis titles and values. Any
David> ideas if such a thing is possible?

Below is the routine I use to create a "ladder" of plots with a
common x-axis. It has several bells and whistles, but here is a simple
example of its use:

```
a = indgen(10)      ; create x-axis
b = randomu(seed,10,4) ; random y-values for 4 plots on common x-axis
ladder,a,b          ; plot them!
```

```
-----

;+
; NAME:    LADDER
; PURPOSE: Graph one or more X-Y plots in separate vertical graphs with a
;          common X-axis to produce a "ladder" of X-Y graphs.
; CATEGORY: Display
; CALLING SEQUENCE: LADDER, A [, B, vinpos=vinpos, vpos=vpos, hplt=hplt,
;                   htotal=htotal, round=round,
;                   noerase=noerase, ppos=ppos]
;
; INPUTS:
;   A = an array or a vector. If a vector, it specifies the values of
;   the X-axis for all the plots, and an input array B should also be
;   provided. If A is an array, then A(*,0) is taken as the vector
;   specifying the X-axis and A(*,i) contains the y-values for the
;   i-th X-Y plot.
;
; OPTIONAL INPUT PARAMETERS:
;   B = an array which must be present if A is a vector.
;   B(*, i) contains the y-values FOR plot i-1.
;
; KEYWORD PARAMETERS:
;   ROUND: By default (round = 0), do not round up y-axis data
;   ranges to convenient values for each graph in the ladder.
;   Calling LADDER with the /ROUND keyword will round up the
;   data range for each graph in the ladder using ROUNDUP.
```

```

;
;
; VINPOS: Vector of dimension nplot+1 (nplot=number of plots)
; specifying the vertical position of each plot in normalized
; coordinates. For plot i, vinpos(i-1) specifies the top and
; vinpos(i) the bottom of the graph in normalized coordinates.
; If VINPOS is not supplied, or is zero, appropriate values
; are determined and returned through the VPOS keyword to
; facilitate subsequent annotation. If VINPOS is set, the PPOS
; option is ignored.
;
;
; VPOS: On output, a vector specifying the vertical position of
; each plot in normalized coordinates. See VINPOS for more info.
;
;
; HPLT: Vector of dimension nplot containing the upper limit for
; each plot. If set, it takes precedence over ROUND.
;
;
; HTOTAL: On output, contains the total height in DATA coordinates
; of the entire ladder of plots. Provided for use in subsequent
; annotation.
;
;
; NOERASE: If set, plot is put on same page as preceding plots.
; Defaults to selecting a new page.
;
;
; PPOS: Vector of four components containing
; [xmin,ymin,xmax,ymax] in normalized coordinates for the
; position of the ladder plot.
;
;
; COMMON BLOCKS: None
; SIDE EFFECTS: Changes display.
; RESTRICTIONS: Requires function ROUNDUP to make convenient y-axis ranges.
; PROCEDURE: Check for consistency in input. Determine height segment
; for each plot, or use VPOS, if supplied. Plot graphs.
; MODIFICATION HISTORY: Written by P.A. Sterne 6/1/1991 (sterne1@llnl.gov)
; PAS: Added optional vinpos. 5/17/93
; PAS: Added option ppos. 5/23/94
;-

```

```

PRO ladder, A, B, vinpos = vinpos, vpos = vpos, hplt = hplt, $
    noerase = noerase, htotal = htotal, round = round, ppos=ppos

```

```

CASE n_params() OF

```

```

1: BEGIN

```

```

    index = size(A)

```

```

    m = index(1)

```

```

    n = index(2)

```

```

    nplot = n-1

```

```

    B = A(*, 1:n-1)

```

```

    A1 = A(*, 0)

```

```

END
2: BEGIN
  x = size(a)
  IF x(0) NE 1 THEN BEGIN ; check: first argument array or vector?
    print, $
    'ladder: first parameter must be a vector of X-coordinates'
    RETURN
  ENDIF

  index = size(B)
  nplot = index(2)
  IF index(1) NE x(1) THEN BEGIN ; check: dimensions A and B match?
    print, $
    'ladder: mismatch in dimensions input parameters:'
    print, 'ladder: A(nx),B(nx,nplot)'
    RETURN
  ENDIF
  A1 = A
END
ENDCASE

xrange = [min (A1), max (A1)]
yst = 1
xtickn = replicate(", 30) ; If there is one plot, the
; ; x tickmarks will appear.
IF nplot GT 1 THEN xtickn = replicate(' ', 30)
; ; Suppress x-tickmarks
yminor = -1 ; Suppress y minor tickmarks

p = fltarr(nplot)
h = fltarr(nplot)
IF (size(hplt))(0) EQ 1 THEN BEGIN ; is hplt defined and a vector?
  hnorm = hplt
ENDIF ELSE BEGIN
  FOR i = 0, nplot-1 DO BEGIN
    h(i) = max (b(*, i))
  ENDFOR
  hnorm = h
  IF KEYWORD_SET(round) THEN BEGIN
    hn = sqrt(h/max(h)) ; magnify relatively smaller sections
    hnorm = roundup(hn*max(h)) ; make nice upper limits
  ENDIF
ENDELSE
htotal = total(hnorm)
FOR i = 0, nplot-1 DO BEGIN ; set the fraction of the total
  p(i) = [hnorm(i)/htotal*0.8] ; graph area that each graph will
ENDFOR ; get, normalized units.
hmax = max(hnorm)

```

```
pmax = max(p)
psc = hmax/pmax
```

```
IF keyword_set(ppos) THEN BEGIN ; scaling to range set by ppos
```

```
  pp0 = [ppos(0), ppos(1), ppos(0), ppos(1)]
```

```
  pp1 = [ppos(2), ppos(3), ppos(2), ppos(3)] - pp0
```

```
  px0 = [0.2, 0.0, 0.8, 0.0]*pp1 + [pp0(0), 0.0, pp0(0), 0.0]
```

```
ENDIF ELSE BEGIN
```

```
  pp0 = [0.0, 0.0, 0.0, 0.0]
```

```
  pp1 = [1.0, 1.0, 1.0, 1.0]
```

```
  px0 = [0.1, 0.0, 0.9, 0.0]*pp1
```

```
;   px0 = [0.0, 0.0, 1.0, 0.0]*pp1
```

```
ENDELSE
```

```
IF KEYWORD_SET(vinpos) THEN BEGIN ; using vinpos
```

```
  pp0 = [0.0, 0.0, 0.0, 0.0]
```

```
  pp1 = [1.0, 1.0, 1.0, 1.0]
```

```
;   px0 = [0.0, 0.0, 1.0, 0.0]*pp1
```

```
  px0 = [0.1, 0.0, 0.9, 0.0]*pp1
```

```
  vpos = vinpos
```

```
ENDIF ELSE BEGIN ; set min,max for each graph (normal coords.)
```

```
  vpos = fltarr(nplot+1)
```

```
  vpos(0) = .9
```

```
  FOR i = 1, nplot DO BEGIN
```

```
    vpos(i) = [vpos(i-1)-p(i-1)]
```

```
  ENDFOR
```

```
  vpos = vpos*pp1(1) + pp0(1)
```

```
ENDELSE
```

```
IF p(0) GE .25 THEN yticks = 0 ; if the graph is large, the ytickmarks
```

```
; ; will be set to the default
```

```
IF (p(0) LT .25) AND (p(0) GT .15) THEN yticks = 2
```

```
; ; However, if the graph is
```

```
; ; smaller, the ytickmarks are set to 2
```

```
IF p(0) LE .15 THEN yticks = 1 ; and if the graph is even smaller,
```

```
; ; the ytickmarks are set to 1
```

```
plot, A1, B(*, 0), pos = px0 + [0.0, vpos(1), 0.0, vpos(0)], $
```

```
  yrange = psc*p(0)*[0, 1], ystyle = yst, xtickname = xtckn, $
```

```
  yticks = yticks, yminor = yminor, noerase = noerase
```

```
IF nplot GT 1 THEN BEGIN
```

```
  FOR i = 1, nplot-1 DO BEGIN
```

```
    IF i EQ nplot-1 THEN xtckn = replicate(", 30)
```

```
    IF p(i) GE .25 THEN yticks = 0
```

```
    IF (p(i) LT .25) AND (p(0) GT .15) THEN yticks = 2
```

```
    IF p(i) LE .15 THEN yticks = 1
```

```
    plot, A1, B(*, i), pos = px0 + [0.0, vpos(i+1), 0.0, vpos(i)], $
```

```
    /noerase, yrange = psc*p(i)*[0, 1], ystyle = yst, $
```

```
    xtickname = xtckn, yticks = yticks, yminor = yminor
```

```

    ENDFOR
  ENDIF
  RETURN
END

```

function roundup, a

```

;
; NAME:
; ROUNDUP
;
; FUNCTION:
;
; Round numbers up to the next highest "convenient" numbers.
; Mantissas for convenient numbers are given in increasing
; order in the array allowv. The numbers in the array can be positive,
; negative or zero. Positive numbers are rounded up, negative numbers
; are rounded down to a larger absolute value, and zero elements are
; left alone.
;
; INPUT:
; a : Array of numbers to be rounded up (down for -ve numbers)
;
; OUTPUT:
; b : Array of convenient output numbers.
;
;
arrayck = size(a)
if(arrayck(0) lt 1 ) then begin ; check that
  print,'function roundup: input must be an array' ; a is an
  return,0 ; array
endif
;
allowv =[1.2,1.5,2,2.5,3,4,5,6,8,10,12] ; output mantissas
incr = 1.05 ; multiply each input mantissa
; ; by incr
;
index = where(a ne 0,count) ; all non-zero elts. of a
if(count eq 0) then begin
  print,'all elements of the array a are zero'
  return,a
endif
;
aI0 = a(index) ; non-zero elts only
signa = aI0/abs(aI0) ; sign of elts.
;
lI0 = alog10(abs(aI0))
expon = fix(lI0)
indx1 = where(lI0 lt 0,count1)

```

```

if count1 ne 0 then expon(indx1) = expon(indx1) - 1 ; correct exponent for
;      ; abs(a) < 1
;
rf10 = incr*10^(l10 - expon)          ; 1 <= rf10 <= 10
n = size(allowv)
b1 = rf10
for i = 0,n(1)-1 do begin      ; work backwards
    indx = where(rf10 lt allowv(n(1)-1-i),count) ; through allowv
    if count gt 0 then b1(indx) = allowv(n(1)-1-i) ; set b1 = allowv
endfor      ; value if
;      ; rf10 < allowv
;
b1 = b1*10^float(expon)*signa ; restore exponent & sign
b = a
b(index) = b1      ; put back zero elts
return,b
end

```

--

```

-----
Philip Sterne |  Sterne@dublin.llnl.gov
Lawrence Livermore National Laboratory |  Phone (510) 422-2510
Livermore, CA 94550 |  Fax (510) 422-7300

```
