Subject: Re: plot legend
Posted by knight on Wed, 26 Aug 1992 13:40:18 GMT
View Forum Message <> Reply to Message

In article <1992Aug25.213818.16912@ll.mit.edu>, I write:
|>
|> Apropos a recent posting, here is a general legend procedure. See the examples
|> in the header for usage.

Attached below is an improvement on yesterday's posting. The new version
allows multiple columns (in vertical format) and multiple rows (in horizontal
format). There's an example in the header under the Optional Output Keyword
section.

Fred

```
;+
; Name:
; legend
; Purpose:
; This procedure makes a legend for a plot.  The legend can contain
; a mixture of symbols, linestyles, and filled polygons (usersym).
; Examples:
; The call:
;  legend,['diamond','asterisk','square'],psym=[4,2,6]
; produces:
;  -----------------
;  |  |
;  |  <> diamond |
;  |  *  asterisk |
;  |  [] square |
;  |  |
;  -----------------
; Each symbol is drawn with a plots command, so they look OK.
; Other examples are given in usage.
; Usage:
; legend,items,linestyle=linestyle ; vertical legend at upper left
; legend,items,psym=psym   ; ditto except using symbols
; legend,items,psym=psym,/horizontal ; horizontal format
; legend,items,psym=psym,box=0  ; sans border
; legend,items,psym=psym,delimiter='=' ; embed an '=' betw psym & text
; legend,items,psym=psym,margin=2  ; 2-character margin
; legend,items,psym=psym,position=pos ; position of legend
; legend,items,psym=psym,number=2  ; plot two symbols, not one
; legend,items,/fill,psym=[8,8,8],colors=[10,20,30]; 3 filled squares
; Inputs:
; items = text for the items in the legend, a string array.
;  You can omit items if you don't want any text labels.
```

```
; Optional Inputs:
; linestyle = array of linestyle numbers
; psym = array of plot symbol numbers.  If psym(i) is negative, then a
;  line connects pts for ith item.  If psym(i) = 8, then the
;  procedure usersym is called with vertices define in the
;  keyword usersym.
; N. B.: Choose either linestyle, psym, neither, or both.  If neither is
;  present, only the text is output.  If both linestyle and
;  psym parameters are present, they both have to have the
;  same number of elements, and normal plot behaviour occurs.
;  By default, if psym is positive, you get one point so there is
;  no connecting line.
; Optional Keywords:
; /help = flag to print header
; /horizontal = flag to make the legend horizontal
; /vertical = flag to make the legend vertical (D=vertical)
; box = flag to include/omit box around the legend (D=include)
; delimiter = embedded character(s) between symbol and text (D=none)
; colors = array of colors for plot symbols/lines (D=!color)
; textcolors = array of colors for text (D=!color)
; margin = margin around text measured in characters and lines
; spacing = line spacing (D=bit more than character height)
; pspacing = psym spacing (D=3 characters)
; charsize = just like !p.charsize for plot labels
; position = normalized coordinates of the upper left of the legend
; number = number of plot symbols to plot or length of line (D=1)
; usersym = 2-D array of vertices, cf. usersym in IDL manual. (D=square)
; /fill = flag to fill the usersym
; Outputs:
; legend to current plot device
; Optional Output Keywords:
; corners = 4-element array, like !p.position, of the normalized
;  coords for the box (even if box=0): [llx,lly,urx,ury].
;  Useful for multi-column or multi-line legends, for example,
;  to make a 2-column legend, you might do the following:
;    c1_items = ['diamond','asterisk','square']
;    c1_psym = [4,2,6]
;    c2_items = ['solid','dashed','dotted']
;    c2_line = [0,2,1]
;    legend,c1_items,psym=c1_psym,corners=c1,box=0
;     legend,c2_items,line=c2_line,corners=c2,box=0,pos=[c1(2),c1( 3)]
;    c = [c1(0)<c2(0),c1(1)<c1(1),c1(2)>c2(2),c1(3)>c1(3)]
;     plots,[c(0),c(0),c(2),c(2),c(0)],[c(1),c(3),c(3),c(1),c(1)], /norm
; Common blocks:
; none
; Procedure:
; If keyword help is set, call doc_library to print header.
; Restrictions:
```

```
; Here are some things that aren't implemented.
; It would be nice to allow data and device coords as well.
; An orientation keyword would allow lines at angles in the legend.
; An array of usersyms would be nice---simple change.
; An order option to interchange symbols and text might be nice.
; Somebody might like double boxes, e.g., with box = 2.
; Another feature might be a continuous bar with ticks and text.
; Side Effects:
; Modification history:
; write, 24-25 Aug 92, F K Knight (knight@ll.mit.edu)
; allow omission of items or omission of both psym and linestyle, add
;   corners keyword to facilitate multi-column legends, 26 Aug 92, FKK
;-
pro legend,help=help,items,linestyle=linestyle,psym=psym $
   ,horizontal=horizontal,vertical=vertical,box=box,margin=marg in $
   ,delimiter=delimiter,spacing=spacing,charsize=charsize,pspac ing=pspacing $
   ,position=position,number=number,colors=colors,textcolors=te xtcolors $
  ,fill=fill,usersym=usersym,corners=corners
;
; =====>> HELP
;
if keyword_set(help) then begin & doc_library,'legend' & return & endif
;
; =====>> SET DEFAULTS
;
ni = n_elements(items)
np = n_elements(psym)
nl = n_elements(linestyle)
n = max([ni,np,nl])
if n eq 0 then message,'No inputs?  For help, type legend,/help.'
if ni eq 0 then begin
  items = replicate('',n)  ; DEFAULT BLANK ARRAY
endif else begin
  szt = size(items)
  if (szt(szt(0)+1) ne 7) then message,'First parameter must be a string array.  For help, type
legend,/help.'
  if ni ne n then message,'Must have number of items equal to '+strtrim(n,2)
endelse
symline = (np ne 0) or (nl ne 0)   ; FLAG TO PLOT SYM/LINE
if n_elements(linestyle) ne n then linestyle = intarr(n); D=SOLID
if n_elements(psym) ne n then psym = intarr(n)  ; D=SOLID

if n_elements(horizontal) eq 0 then begin   ; D=VERTICAL
  if n_elements(vertical) eq 0 then vertical = 1
endif else begin
  if n_elements(vertical) eq 0 then vertical = not horizontal
endelse
```

```
if n_elements(box) eq 0 then box = 1
if n_elements(margin) eq 0 then margin = 1
if n_elements(delimiter) eq 0 then delimiter = ''
if n_elements(charsize) eq 0 then charsize = 1
if n_elements(spacing) eq 0 then spacing = 1.2
if n_elements(pspacing) eq 0 then pspacing = 3
xspacing = !d.x_ch_size/float(!d.x_size) * (spacing > charsize)
yspacing = !d.y_ch_size/float(!d.y_size) * (spacing > charsize)
if !x.window(0) eq !x.window(1) then begin
  plot,/nodata,xstyle=4,ystyle=4,[0],/noerase
endif
; next line takes care of weirdness with small windows
pos =  [min(!x.window),min(!y.window),max(!x.window),max(!y.window) ]
if n_elements(position) eq 0 then position = [pos(0),pos(3)] + [xspacing,-yspacing]
if n_elements(number) eq 0 then number = 1
if n_elements(colors) eq 0 then colors = !color + intarr(n)
if n_elements(textcolors) eq 0 then textcolors = !color + intarr(n)
fill = keyword_set(fill)
if n_elements(usersym) eq 0 then usersym = [[0,0],[0,1],[1,1],[1,0]]-0.5
;
; =====>> INITIALIZE POSITIONS
;
yoff = 0.2*yspacing
maxwidth = 0     ; SAVED WIDTH FOR DRAWING BOX
x0 = position(0) + (margin)*xspacing  ; INITIAL X & Y POSITIONS
y0 = position(1) - (margin-0.5)*yspacing
y = y0      ; STARTING X & Y POSITIONS
x = x0
if vertical then begin    ; CALC OFFSET FOR TEXT START
  xt = 0     ; DEFAULT X VALUE
  for i = 0,n-1 do begin
    if psym(i) eq 0 then num = (number + 1) > 3 else num = number
    if psym(i) lt 0 then num = number > 2 ; TO SHOW CONNECTING LINE
    if psym(i) eq 0 then expand = 1 else expand = 2
    xt = (expand*pspacing*(num-1)*xspacing) > xt
    endfor
endif ; NOW xt IS AN X OFFSET TO ALIGN ALL TEXT ENTRIES
;
; =====>> OUTPUT TEXT FOR LEGEND, ITEM BY ITEM
; =====>> FOR EACH ITEM, PLACE SYM/LINE, THEN DELIMITER,
; =====>> THEN TEXT---UPDATING X & Y POSITIONS EACH TIME.
;
for i = 0,n-1 do begin
  if vertical then x = x0 else y = y0  ; RESET EITHER X OR Y
  x = x + xspacing
  y = y - yspacing
  if psym(i) eq 0 then num = (number + 1) > 3 else num = number
  if psym(i) lt 0 then num = number > 2  ; TO SHOW CONNECTING LINE
```

```
  if psym(i) eq 0 then expand = 1 else expand = 2
  xp = x + expand*pspacing*indgen(num)*xspacing
  yp = y + yoff + intarr(num)
  if psym(i) eq 0 then begin
    xp = [min(xp),max(xp)]   ; TO EXPOSE LINESTYLES
    yp = [min(yp),max(yp)]   ; DITTO
    endif
  if psym(i) eq 8 then usersym,usersym*charsize,fill=fill,color=colors(i)
  if symline then plots,xp,yp,color=colors(i),/normal $
    ,linestyle=linestyle(i),psym=psym(i),symsize=charsize
  if vertical then x = x + xt else x = max(xp)
  if symline then x = x + xspacing
   xyouts,x,y,delimiter,width=width,/norm,color=textcolors(i),s ize=charsize
  x = x + width
  if width gt 0 then x = x + xspacing
   xyouts,x,y,items(i),width=width,/norm,color=textcolors(i),si ze=charsize
  x = x + width
  if not vertical and (i lt (n-1)) then x = x+2*xspacing; ADD INTER-ITEM SPACE
  maxwidth = (x + xspacing*margin) > maxwidth ; UPDATE MAXIMUM X
  endfor
;
; =====>> OUTPUT BORDER
;
x = position(0)
y = position(1)
if vertical then bottom = n else bottom = 1
ywidth = - (2*margin+bottom-0.5)*yspacing
corners = [x,y+ywidth,maxwidth,y]
if box then plots,[x,maxwidth,maxwidth,x,x],y + [0,0,ywidth,ywidth,0],/norm
return
end
```

--
=Fred Knight (knight@ll.mit.edu) (617) 981-2027
 C-483\\MIT Lincoln Laboratory\\244 Wood Street\\Lexington, MA 02173