Subject: legend (new version and test)
Posted by knight on Mon, 31 Aug 1992 13:46:04 GMT
View Forum Message <> Reply to Message

Last week I posted two versions of a procedure to add a legend to a plot.
Here's a third: slightly improved and with a few more options. Also, I wrote
a test procedure that shows legend's capabilities.

Before I include the procedures, I have one general comment. In the past few
days there have been postings of routines that lack headers, that is,
something that describes the routine in detail like the recommended *standard*
header promoted by RSI, For example,

```
;+
; Name:
; legend
; Purpose:
; This procedure makes a legend for a plot.  The legend can contain
; a mixture of symbols, linestyles, Hershey characters (vectorfont),
; and filled polygons (usersym).

...
;-
```

By using this type of header you not only document the use of your routine,
but you can also provide a straightforward on-line help. For example, the
user can type

```
 legend,/help
```

and get the header printed out with more. The reason is that legend
contains the line:

```
;
; =====>> HELP
;
if keyword_set(help) then begin & doc_library,'legend' & return & endif
```

Using this method gives the user a way to remind herself about the routine.
Also, the IDL procedure xdl will pick up this header as long as the directory
containing the code is in the !path variable. In summary, I
recommend using a header and hope each posted routine includes one.

Now for the third and final version of legend. THere are two procedures:

```
 legend.pro ; creates legend
 legendtest.pro ; shows examples
```

---------- cut here ----------

```
;+
; Name:
; legend
; Purpose:
; This procedure makes a legend for a plot.  The legend can contain
; a mixture of symbols, linestyles, Hershey characters (vectorfont),
; 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.
;  For example, items = ['diamond','asterisk','square'].
; Optional Inputs:
; linestyle = array of linestyle numbers  If linestyle(i) < 0, then omit
;  ith symbol or line to allow a multi-line entry.
; 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.
; vectorfont = vector-drawn characters for the sym/line column, e.g.,
```

```
;  ['!9B!3','!9C!3','!9D!3'] produces an open square, a checkmark,
;  and a partial derivative, which might have accompanying items
;  ['BOX','CHECK','PARTIAL DERIVATIVE'].  If vectorfont(i) = '',
;  then plots is called to make a symbol or a line, but if
;  vectorfont(i) is a non-null string, then xyouts is called.
;  There is no check that !p.font is set properly, e.g., -1 for
;  X and 0 for PostScript.  This can produce an error, e.g., use
;  !20 with PostScript and !p.font=0, but allows use of Hershey
;  *AND* PostScript fonts together.
; 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)<c2(1),c1(2)>c2(2),c1(3)>c2(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.
; - There are no guards to avoid writing outside the plot area.
; - There is no provision for multi-line text, e.g., '1st line!c2nd line'
;   Sensing !c would be easy, but !c isn't implemented for PostScript.
;   A better way might be to simply output the 2nd line as another item
;   but without any accompanying symbol or linestyle.  A flag to omit
;   the symbol and linestyle is linestyle(i) = -1.
; 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, improve place-
;   ment of symbols and text, add guards for unequal size, 26 Aug 92, FKK
; add linestyle(i)=-1 to suppress a single symbol/line, 27 Aug 92, FKK
; add keyword vectorfont to allow characters in the sym/line column,
;   28 Aug 92, FKK
;-
pro  legend,help=help,items,linestyle=linestyle,psym=psym,vectorf ont=vectorfont $
   ,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 FOR SYMBOLS, LINESTYLES, AND ITEMS.
;
ni = n_elements(items)
np = n_elements(psym)
nl = n_elements(linestyle)
nv = n_elements(vectorfont)
n = max([ni,np,nl,nv])     ; NUMBER OF ENTRIES
strn = strtrim(n,2)     ; FOR ERROR MESSAGES
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 '+strn
endelse
symline = (np ne 0) or (nl ne 0)   ; FLAG TO PLOT SYM/LINE
```

```
if (np ne 0) and (np ne n) then message,'Must have 0 or '+strn+' elements in psym array.'
if (nl ne 0) and (nl ne n) then message,'Must have 0 or '+strn+' elements in linestyle array.'
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(vectorfont) ne n then vectorfont = replicate('',n)
;
; =====>> CHOOSE VERTICAL OR HORIZONTAL ORIENTATION.
;
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
;
; =====>> SET DEFAULTS FOR OTHER OPTIONS.
;
if n_elements(box) eq 0 then box = 1
if n_elements(margin) eq 0 then margin = 0.5
if n_elements(delimiter) eq 0 then delimiter = ''
if n_elements(charsize) eq 0 then charsize = !p.charsize
if 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 = 2*[[0,0],[0,1],[1,1],[1,0]]-1
;
; =====>> INITIALIZE POSITIONS
;
yoff = 0.25*yspacing    ; VERT. OFFSET FOR SYM/LINE.
maxx = 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.
; =====>> THERE ARE A NUMBER OF EXCEPTIONS DONE WITH IF STATEMENTS.
;
for i = 0,n-1 do begin
  if vertical then x = x0 else y = y0  ; RESET EITHER X OR Y
  x = x + xspacing    ; UPDATE X & Y POSITIONS
  y = y - yspacing
  if (psym(i) eq 0) and (vectorfont(i) eq '') 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
  if (psym(i) gt 0) and (num eq 1) and vertical then xp = x + xt/2.
  yp = y + intarr(num)
  if vectorfont(i) eq '' then yp = yp + yoff
  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,fill=fill,color=colors(i)
  if vectorfont(i) ne '' then begin
    if (num eq 1) and vertical then xp = x + xt/2
    xyouts,xp,yp,vectorfont(i),width=width,color=colors(i) $
      ,size=charsize,align=0.5,/norm
  endif else begin
    if symline and (linestyle(i) ge 0) then plots,xp,yp,color=colors(i) $
       ,/normal,linestyle=linestyle(i),psym=psym(i),symsize=charsiz e
  endelse
  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 + 0.5*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
  maxx = (x + xspacing*margin) > maxx  ; 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,maxx,y]
if box then plots,[x,maxx,maxx,x,x],y + [0,0,ywidth,ywidth,0],/norm
return
end
------------ cut here -------------------------
;+
; Name:
; legendtest
; Purpose:
; Test the legend procedure.
; Usage:
; .run legendtest
; Inputs:
; none
; Optional Inputs:
; none
; Keywords:
; none
; Outputs:
; legends of note
; Common blocks:
; none
; Procedure:
; Side Effects:
; Sets !20 font to symbol if PostScript and !p.font=0.
; Restrictions:
; With the vectorfont test, you'll get different results for PostScript
; depending on the value of !p.font.
; Modification history:
; write, 27 Aug 92, F.K.Knight (knight@ll.mit.edu)
;-
if (!d.name eq 'PS') and (!p.font eq 0) then device,/Symbol,font_index=20
items = ['diamond','asterisk','square']
psym = [4,2,6]
lineitems = ['solid','dotted','DASHED']
linestyle = [0,1,2]
citems = 'color '+strtrim(string(indgen(8)),2)
colors = 15*indgen(8)+50
z = ['legend,items,psym=[4,2,6]' $
 ,'legend,lineitems,linestyle=linestyle' $
 ,'legend,items,psym=psym,/horizontal,chars=1.5 ; horizontal format' $
  ,'legend,[items,lineitems],psym=[psym,0,0,0],line=[0,0,0,lin estyle],box=0  ; sans border' $
  ,'legend,items,psym=psym,margin=1,spacing=2,char=2,delimiter = "="; delimiter & larger margin'
```

```
$
  ,'legend,lineitems,line=linestyle,pos=[.3,.5],char=3,number= 4 ; position of legend' $
 ,'legend,items,psym=-psym,number=2,line=linestyle; plot two symbols, not one' $
  ,'legend,citems,/fill,psym=8+intarr(8),colors=colors,char=2; 3 filled squares' $
  ,'legend,[citems(0:4),lineitems],/fill,psym=[8+intarr(5),0*p
sym],line=[intarr(5),linestyle],colors=colors,char=2; 3 filled squares' $
 ,''legend,['Absurd','Sun Lover','Lucky Lady','Fishtail
Palm'],vector=['ab!9r!3','!9nu!3','!9Wf!3','!9cN!20K!3'],cha rsize=2,pos=[.1,.5],psp=3 " $
 ]
prompt = 'Hit return to continue:'
for i = 0,n_elements(z) - 1 do begin
  erase
  stat = execute(z(i))
  xyouts,.01,.15,'COMMAND TO MAKE LEGEND:',charsize=1.7,/norm
  xyouts,.01,.05,z(i),/norm,charsize=1.2
  print,prompt,format='($,a)'
  a = get_kbrd(1)
  print
  endfor
erase
!p.charsize=2
c1_items =  ['Plus','Asterick','Period','Diamond','Triangle','Square','X ']
c1_psym = indgen(7)+1
c2_items = ['Solid','Dotted','Dashed','Dash Dot','Dash Dot Dot Dot','Long Dashes']
c2_line = indgen(6)
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)<c2(1),c1(2)>c2(2),c1(3)>c2(3)]
 plots,[c(0),c(0),c(2),c(2),c(0)],[c(1),c(3),c(3),c(1),c(1)], /norm
!p.charsize=0
xyouts,.01,.05,'Multiple columns---type "legend,/help" for details.',/norm,charsize=1.2
end---------- cut here ------------------------
--
=Fred Knight (knight@ll.mit.edu) (617) 981-2027
 C-483\\MIT Lincoln Laboratory\\244 Wood Street\\Lexington, MA  02173
```