Subject: multipanel
Posted by Martin Schultz on Tue, 23 Mar 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Hi all,

please find below a routine designed to ease one's life with !p.multi. It was inspired by David F's new tvimage routine which "honors" the !p.multi keyword. You can use multipanel.pro to:

* get the position vector for the next plot to be made (regardless of !p.multi)
* set up a multi-panel plot specifying either the number of rows and cols or
the number of plots per page
* advance to the next (or to any) panel on the page
* turn !p.multi off

There are MARGIN and OMARGIN options which accept 1,2 or 4 values in normalized coordinates, and (of course) a /NOERASE keyword. If you set up a page with say 5 panels per page, the screen (or page) will actually be erased after 5 plots although 6 would fit onto the page and "plain" !p.multi would do all 6.

Try it out! It's free ;-)

Martin.

--
---------- -------
Dr. Martin Schultz
Department for Engineering&Applied Sciences, Harvard University
109 Pierce Hall, 29 Oxford St., Cambridge, MA-02138, USA

phone: (617)-496-8318
fax : (617)-495-4551

e-mail: mgs@io.harvard.edu
Internet-homepage: http://www-as.harvard.edu/people/staff/mgs/
---------- -------
;---------- --
; Id
;+
; NAME:
; MULTIPANEL
;

```
; PURPOSE:
;       This routine provides an easy-to-use interface to
;       IDL's !p.multi mechanism and it can be used
;       to retrieve the position vector of the next plot
;       in line (even if !p.multi is not set). You can
;       specify plot margins and a page margin in normalized
;       coordinates.
;         For multi-panel plots, multipanel works in three
;       stages (somewhat similar to XInterAnimate):
;       1) set up a multi-panel page by specifying the number
;         of rows and columns or the maximum number of plots
;         on the page.
;       2) advance to the next plot position and return that
;         position. If the page was filled (i.e. the maximum
;         number of plots (advance steps) has been made, it
;         will be erased (unless /NOERASE is set).
;       3) turn the multi-panel environment off and reset margin
;         values.
;       Subsequent PLOT, MAP_SET, etc. commands should use the
;       POSITION vector returned from this routine and they should
;       use the /NOERASE keyword.
;
; CATEGORY:
;       Plotting tools
;
; CALLING SEQUENCE:
;       Get position of next plot
;       MULTIPANEL,position=p
;
;       Setting up multi-panel pots
;       MULTIPANEL,[rows=r,cols=c | nplots=n] [,options]
;
;       Advance to the next panel (and get its position)
;       MULTIPANEL,/advance [,position=p] [,options]
;
;       Reset multi-panel environment to start over at the first panel
;       MULTIPANEL,/reset   [,options]
;
;       Turn multi-panel environment off (reset !p.multi and some other
;       parameters)
;       MULTIPANEL,/OFF
;
; INPUTS:
;    Optionally you can give the number of plots (keyword NPLOTS) as
;    a parameter instead. Th euse of a parameter will override the
;    keyword.
;
; KEYWORD PARAMETERS:
```

```
;       General keywords (honored under all circumstances)
;        ------------------------------------------------------------ -------
;       POSITION -> Returns a 4-element vector with the position of
;          the next plot in the line. As always this vector contains
;          [X0,Y0,X1,Y1].
;
;       MARGIN -> specify a margin around the plot in normalized
;          coordinates. This keyword does not change any IDL
;          system variables and will thus only become "visible"
;          if you use the POSITION returned by MULTIPANEL in subsequent plot
;          commands.
;          MARGIN can either be one value which will be applied to
;          all four margins, or a 2-element vector which results in
;          equal values for the left and right and equal values for
;          the bottom and top margins, or a 4-element vector with
;          [left,bottom,right,top].
;
;       OMARGIN -> specify a page margin around all panels in normalized
;          coordinates. Works like MARGIN.
;
;       Note that you can change the values of MARGIN and OMARGIN after
;       you set up your plot page. The values given with MARGIN and OMARGIN
;       remain active until you specify a new one or use the /OFF keyword.
;
;       /NOERASE -> This keyword prevents erasing the screen (or page)
;          when setting a multi-panel environment or after a page was
;          filled. NOERASE is automatically turned ON when the /OFF
;          keyword is given.
;
;
;       Informational keywords:
;        ------------------------------------------------------------ -------
;
;       FIRSTPANEL -> returns 1 if the current plotting panel is
;          the first one on a page
;
;       LASTPANEL -> returns 1 if the current plotting panel is the last
;          on a page
;
;
;       Setting up a multi-panel page:
;        ------------------------------------------------------------ -------
;       ROWS, COLS -> specify the number of rows and columns
;          you want on your page. If one is specified, the
;          other one must be given as well. Alternatively,
;          you can use the NPLOTS keyword.
;
;       NPLOTS -> maximum number of plots on one page. The
```

```
;          number of rows and columns is automatically computed
;          so that the plots "approach a square" and they are
;          returned in the ROWS and COLS keywords. Setting
;          NPLOTS to a ROWS*COLS value is equivalent to using
;          ROWS and COLS. If you specify an "uneven" number
;          (e.g. 5), multipanel,/advance will erase the page
;          after 5 plots instead of 6.
;
;       /PORTRAIT -> Normally, ROWS tends to be larger than COLS
;          when NPLOTS is used (e.g. 12 gives 4 rows and 3 cols).
;          Use this keyword to revert this behaviour.
;
;       /LANDSCAPE -> Make ROWS larger than COLS if necessary. This
;          is the default. Thi skeyword is actually unnecessary and
;          was introduced for purely aesthetic reasons (symmetry).
;
;
;       Advance to the next plot:
;        ------------------------------------------------------------ -------
;       /ADVANCE -> this keyword issues a hidden plot command to find
;          out the position of the next plot to be made. The position
;          is then returned in the POSITION keyword. The value of
;          !P.MULTI[0] is increased afterwards so that you can issue
;          your plot command without explicitely specifying the
;          position or /NOERASE. When the maximum number of plots set
;          with NPLOTS is reached, MULTIPANEL,/ADVANCE will erase the
;          screen and reset the plot position to the upper left corner.
;
;       CURPLOT -> use this keyword to advance to a specific plot position
;          on the page. If you specify 0, the screen will be erased.
;          CURPLOT also returns the current plot number. If you don't
;          want to set the plot number but just query it, you must pass
;          an undefined variable (see procedure UNDEFINE.PRO).
;
;
;       Reset the plot position (and erase the screen)
;        ------------------------------------------------------------ -------
;       /RESET -> does just this.
;
;
;       Turn multi-panel environment off
;        ------------------------------------------------------------ -------
;       /OFF -> Overrides all other keywords. Resets !p.multi to 0.
;
;
; OUTPUTS:
;       none.
;
```

```
; SUBROUTINES:
;       function GET__PLOTPOSITION() issues a dummy plot command
;           and returns the plot position from the ![xy].window
;           variables.
;
; REQUIREMENTS:
;       none. (example uses RECTANGLE.PRO)
;
; NOTES:
;       Side effect: Opens a window in standard size if none was open
;       before. This happens because the next plot position is
;       determined by issuing a hidden (or dummy) plot command with
;       no visible output.
;
;       Make sure to use POSITION=P,/NOERASE with all your PLOT, MAP_SET
;       CONTOUR, etc. commands. In fact, you don't _need_ /NOERASE for
;       PLOT, but it guarantees consistent behaviour with MAP_SET etc.)
;       A PLOT command without /NOERASE will advance to the next panel
;       and thereby interfere with MULTIPANEL's plot counter.
;
;       If you don't use MARGIN and OMARGIN, the values in ![XY].margin
;       ![XY].omargin will take effect.
;
;       A common block is used to store the current plot number and
;       margin information. This limits the use to one window at a time
;       (although you can save all parameters elsewhere and supply them
;       to the routine on a window-by-window basis).
;
; EXAMPLES:
;       ; just retrieve the position of the next plot
;       ;  ------------------------------------------------------------ ---
;       MULTIPANEL,position=p
;       PLOT,findgen(10),color=1,position=p,/noerase
;
;       ; same thing but use a specific margin (2-element form)
;       ;  ------------------------------------------------------------ ---
;       MULTIPANEL,margin=[0.1,0.05],position=p
;       PLOT,findgen(10),color=1,position=p,/noerase
;
;       ; (if you want to see both previous plots together use
;       ' /NOERASE with MULTIPANEL)
;
;       ; multi-panel use:
;       ;  ------------------------------------------------------------ ---
;       MULTIPANEL,NPlots=5                 ; set up for 5 plots
;       for i=0,4 do begin
;         MULTIPANEL,Position=p,margin=0.04    ; get current plot position
;         plot,findgen(10),color=1,position=p,/noerase   ; plot
```

```
;          MULTIPANEL,/Advance,/NoErase ; go to next panel
;          ; note that the screen would be erased after the last plot
;          ; without the /NoErase keyword.
;       endfor
;       ; now let's draw a frame around everything
;       MULTIPANEL,/OFF,omargin=0.01,margin=0.,position=p,/noerase
;       Rectangle,p,xvec,yvec                ; <<< uses RECTANGLE.PRO !!
;       plots,xvec,yvec,color=1,/norm
;
; MODIFICATION HISTORY:
;       mgs, 19 Mar 1999: VERSION 1.00
;       mgs, 22 Mar 1999: - improved documentation, changed OMARGIN
;                to accept normalized coordinates.
;                      - position now also returned if turned OFF
;                      - added FIRSTPANEL and LASTPANEL keywords
;                      - allow NPLOTS to be specified as parameter
;
;-
; Copyright (C) 1999, Martin Schultz, Harvard University
; This software is provided as is without any warranty
; whatsoever. It may be freely used, copied or distributed
; for non-commercial purposes. This copyright notice must be
; kept with any copy of this software. If this software shall
; be used commercially or sold as part of a larger package,
; please contact the author to arrange payment.
; Bugs and comments should be directed to mgs@io.harvard.edu
; with subject "IDL routine multipanel"
 ;-------------------------------------------------------------- --


function get__plotposition,noerase=noerase,advance=advance

COMMON PLOT__MARGINS, activemargin, margingiven, pg_, $
                maxplots, curplot


   ; increase current plot number except if noadvance was given

   if (keyword_set(advance)) then curplot = ( curplot + 1 ) > 0
   if (curplot ge maxplots) then curplot = 0

   ; set new value of !p.multi[0] according to curplot
   ; this has to be done in order to achieve the same behaviour
   ; with map_set as with plot
   ; note that !p.multi will not be zero at any time, thus the
   ; next plot will not erase the screen automatically.

   maxrc = ( !p.multi[1]*!p.multi[2] ) > 1
```

```
   !p.multi[0] = maxrc - curplot

   ; save this value of !p.multi[0]
   save_p0 = !p.multi[0]

;  if (not keyword_set(advance)) then !p.multi[0] = !p.multi[0] + 1
;  if (!p.multi[0] gt maxplots) then !p.multi[0] = maxrc

; print,'get__.... : curplot,!p.multi[0] = ',curplot,!p.multi[0]

   ; erase screen or page manually if necessary
   if (not keyword_set(noerase) AND curplot eq 0) then erase

   ; create an empty dummy plot and return position
   plot,findgen(10),/nodata,xstyle=4,ystyle=4

   ; reset !p.multi to value it had before dummy plot
   !p.multi[0] = save_p0

   ; get plot window position :
   ; 1. extract it from the window coordinates
   result = [ !x.window[0], !y.window[0], $
              !x.window[1], !y.window[1]  ]

   ; apply margin correction
   result = result+activemargin

   ; shrink to fit into page size
   result[[0,2]] = result[[0,2]]*pg_.wx + pg_.ox
   result[[1,3]] = result[[1,3]]*pg_.wy + pg_.oy

   return, result
end


; $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$


pro multipanel,ParNPlots,   $
       rows=rows, cols=cols, margin=margin, omargin=omargin, $
       nplots=nplots, portrait=portrait, landscape=landscape, $
       position=position,  $
       advance=advance,  $
       reset=reset, noerase=noerase,  $
       curplot=tcurplot,firstpanel=firstpanel,lastpanel=lastpanel,  $
       off=off
```

```
COMMON PLOT__MARGINS, activemargin, margingiven, pg_,  $
                maxplots, curplot

  if (n_elements(activemargin) eq 0) then begin
    pg_ = { wx:1., wy:1., ox:0., oy:0. }   ; page width and offset
    activemargin = [ 0., 0., 0., 0. ]
    activeomargin = [ 0., 0., 0., 0. ]
    margingiven = 0    ; no user-supplied margin
                       ; extra variable , because user may supply
                       ; [0,0,0,0] as margin. For omargin, 0. is the
                       ; default.
    maxplots = 1
    curplot = 0
  endif



  ;  ------------------------------------------------------------ ----------
  ; highest significance is given to the keyword that turns !p.multi off
  ;  ------------------------------------------------------------ ----------

  if (keyword_set(OFF)) then begin
    !p.multi = 0

    ; reset margin values
    pg_ = { wx:1., wy:1., ox:0., oy:0. }   ; page width and offset
    activemargin = [ 0., 0., 0., 0. ]
    margingiven = 0
    maxplots = 1
    curplot = 0

    Advance = 0     ; we don't want to advance any further
    NoErase = 1     ; do not erase previous work
  endif



  ;  ------------------------------------------------------------ ----------
  ; now let's deal with keywords that turn !p.multi ON
  ; we need either rows and cols or nplots
  ; nplots is given priority since it can then return the computed
  ; number of rows and columns in the respective variables
  ;  ------------------------------------------------------------ ----------

  ; set default for position
  position = [ 0., 0., 1., 1. ]


  IsTurnedOn = 0
```

```
if (n_elements(Advance) eq 0) then $ ; trick to find out whether turned off
   Advance = keyword_set(Advance)

if (n_params() gt 0) then nplots = fix(ParNPlots[0])
if (n_elements(nplots) gt 0) then begin
   if (nplots le 0) then begin
      message,'NPLOTS must be positive!',/Continue
      return
   endif

   cols = ( fix( sqrt(nplots[0]-1) ) + 1 ) > 1
   rows = ( fix( (nplots[0]-1)/cols ) + 1 ) > 1

   ; switch rows and cols if portrait is selected
   ; (landscape is ignored and just included for
   ; cosmetic reasons)
   if (keyword_set(portrait)) then begin
      tmp = rows
      rows  = cols
      cols = tmp
   endif

   maxplots = nplots > 1

   IsTurnedOn = 1

endif else $
if (n_elements(rows) gt 0 AND n_elements(cols) gt 0) then begin

   maxplots = rows*cols > 1

   IsTurnedOn = 1
endif

if (IsTurnedOn) then begin
   !p.multi = [ 0, cols, rows ]

   ; reset margin values (may be overwritten by explicit margin values)
   pg_ = { wx:1., wy:1., ox:0., oy:0. }   ; page width and offset
   activemargin = [ 0., 0., 0., 0. ]
   margingiven = 0
   curplot = 0

   ; prevent advancing to second frame
   Advance = 0
endif
```

```
;  ---------------------------------------------------------- ----------
; does the user want to reset !p.multi to the first plot?
;  ---------------------------------------------------------- ----------

IsReset = 0
if (IsTurnedOn eq 0 AND keyword_set(Reset)) then begin
   curplot = 0
   IsReset = 1

   ; prevent advancing to second frame
   Advance = 0
endif



;  ---------------------------------------------------------- ----------
; get current number of rows and columns from !p.multi
; This is used to limit the margin range and to determine whether
; the maximum number of plots has been reached
;  ---------------------------------------------------------- ----------

nrows = !p.multi[1] > 1
ncols = !p.multi[2] > 1

;  ---------------------------------------------------------- ----------
; Handle the margin keywords. If it is a single value, use it for
; all four margins. Two values indicate equal x and equal y margins.
; Four values can be stored directly.
; Invert right and top margin values in activemargin so we just need
; to add it onto position.
; Make sure that left is left, and right is right, etc.
;  ---------------------------------------------------------- ----------

if (n_elements(omargin) gt 0) then begin
   tmargin = omargin
   if (n_elements(omargin) eq 1) then tmargin = replicate(omargin,4)
   if (n_elements(omargin) eq 2) then tmargin = [ omargin, omargin ]
   if (n_elements(tmargin) ne 4) then begin
      message,'OMARGIN must have 1, 2, or 4 elements!',/Continue
      return
   endif
   tmargin = abs(tmargin)

   ; Limit OMARGIN to a total of 0.9 at most
   ; issue a warning if they exceed a total of 0.5
   if (tmargin[0]+tmargin[2] gt 0.5) then $
      message,'LEFT/RIGHT omargins exceed 0.5!',/INFO
   if (tmargin[0]+tmargin[2] gt 0.9) then $
```

```
      tmargin[[0,2]] = 0.25

    if (tmargin[1]+tmargin[3] gt 0.5) then $
      message,'LEFT/RIGHT omargins exceed 0.5!',/INFO
    if (tmargin[1]+tmargin[3] gt 0.9) then $
      tmargin[[1,3]] = 0.25

    ; Translate tmargin into "page" parameters of common block
    pg_.wx = 1. - (tmargin[0]+tmargin[2])
    pg_.wy = 1. - (tmargin[1]+tmargin[3])
    pg_.ox = tmargin[0]
    pg_.oy = tmargin[1]

  endif


  if (n_elements(margin) gt 0) then begin
    tmargin = margin
    if (n_elements(margin) eq 1) then tmargin = replicate(margin,4)
    if (n_elements(margin) eq 2) then tmargin = [ margin, margin ]
    if (n_elements(tmargin) ne 4) then begin
      message,'MARGIN must have 1, 2, or 4 elements!',/Continue
      return
    endif
    tmargin = abs(tmargin)

    ; make sure plot has still wiggle room
    ; (note that margin is applied to each plot!)
    if (nrows*(tmargin[0]+tmargin[2]) ge 1.) then begin
      tmargin[[0,2]] = 0.5/nrows - 0.01
      message,'LEFT/RIGHT margins too large! Set to maximum of'  $
          + string(tmargin[0],format='(f8.3)'),/Continue
    endif
    if (ncols*(tmargin[1]+tmargin[3]) ge 1.) then begin
      tmargin[[1,3]] = 0.5/ncols - 0.01
      message,'BOTTOM/TOP margins too large! Set to maximum of'  $
          + string(tmargin[1],format='(f8.3)'),/Continue
    endif

    ; invert right and top values
    tmargin[[2,3]] = -tmargin[[2,3]]

    ; store in common block for later use
    activemargin = tmargin
    margingiven = 1
  endif
```

```
  ;  ---------------------------------------------------------- ----------
  ; Set plot position if specifically requested
  ; Range checking done automatically in get__plotposition
  ;  ---------------------------------------------------------- ----------

  if (n_elements(tcurplot) gt 0) then $
     curplot = fix(tcurplot)

  ;  ---------------------------------------------------------- ----------
  ; Return position of next plot
  ; This automatically advances the plot position unless the
  ; NoAdvance keyword is set!!
  ;  ---------------------------------------------------------- ----------

  ; set margin values in !x and !y to zero only if a user
  ; margin was given
  if (margingiven) then begin
     !x.margin = [0,0]
     !y.margin = [0,0]
  endif

  ; determine position of next plot
  position = get__plotposition(noerase=noerase,Advance=Advance)

  ; return current plot number
  tcurplot = curplot
  firstpanel = (curplot eq 0)
  lastpanel = (curplot eq (maxplots-1) )

  return
end
```

File Attachments

1) multipanel.pro, downloaded 89 times