Subject: Re: Text processing on plots
Posted by gunter on Fri, 14 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Jeff Hicke (jhicke@al.noaa.gov) wrote:
: Hello,
:
: I am interested in whether anyone has any IDL routines which allows one
: to overplot a variable length text string in the manner of a word
: processor. In other words, if I define a box (or width of a box), then
: the routine overplots the text within this box's width. If it is longer
: than one line, it will wrap to a new line. I'm looking at adding figure
: captions to IDL plots. Can anyone help?
:
: Thanks,
: Jeff Hicke
: NOAA Aeronomy Lab
: Boulder CO

Strange, I was just working on the same thing (well, almost.) I wanted to add text to a plot interactively. I've appended two routines at the end of this post for doing such things. The first is named add_text.pro and is a widget-based routine which prompts for the text to be added (multi-lines allowed) and includes a slider for controling the text-size. Clicking the 'OK' button starts the second routine, place.pro, which moves a ghost-box around the plot window under control of the mouse. The ghost-box is set to be the size of the text that will be added. Clicking any of the mouse buttons will place the text at the cursor location. The the idea behind place.pro is stolen from curbox.pro from the ESRG routines.

Note: I've just written these codes this morning and they are sparsely documented. Also, several checks should be implemented such as making sure that you are adding text to the correct plot window (if more than one is open), etc.

Let me know if there are major errors with these. Good luck!

--
david gunter
http://www.mcs.anl.gov/people/gunter/

"When you are a Bear of Very Little Brain, and you Think of Things, you find sometimes that a Thing which seemed very Thingish inside you is quite different when it gets out into the open and has other people looking at it."
- A.A. Milne, "The House At Pooh Corner"

```
 ------------------------------------------------------------ ------------------
pro add_text_event, event

widget_control, event.top, get_uvalue=info

eventname = tag_names(event, /STRUCTURE_NAME)

if eventname EQ 'WIDGET_BUTTON' then begin
   widget_control, event.id, get_value=button

   case button of

      ' Cancel ': begin
         widget_control, event.top, /destroy
      end

      ' OK ': begin
         widget_control, info.text, get_value=text
         n_lines = n_elements(text)

         if n_lines LT 1 then return  ; No text was entered.

; Remove NULL elements if at the end

         if text(n_lines-1) EQ '' then begin
            text = text(0:n_lines-2)
            n_lines=n_lines-1
         endif

; Make sure that some text has been entered
         text_idx = where(text NE '', hits)
         if hits GT 0 then begin

; Concatenate all lines, putting !c's for cariage returns.
            all_text=''
            for i=0, n_lines-1 do $
              all_text = all_text+text(i)+'!c'

; Get the value of the slider.
            widget_control, info.char_sizer, get_value=char_size
; Put the slider in the text window. (I didn't like the way the slider
; displayed it's own value.)
            widget_control, info.char_size, set_value=char_size

; Determine the height (in pixels) of a string.
; (Thanks go out to David Fanning!)
            box_height = !D.Y_CH_SIZE * n_lines * char_size * 1.1
```

```
; Convert to /NORMAL coordinates
            box_height = float(box_height)/float(!D.Y_VSIZE)

; The above fails for string widths because of spacing.  Instead use xyouts
; with charsize=-1 to get the width returned in /NORMAL coordinates.  No
; output is actually displayed. (Thanks again David!)
            xyouts, 0, 0, all_text, width=box_width, size=char_size, $
              charsize=-1
; Assign coordinates for interactive box display (in /NORMAL coordinates).
            left = 0.1
            right = left + box_width
            bottom = 0.9
            top =  bottom + box_height

; Call the interactive display routine.
            place, left, right, bottom, top, /UPPER_LEFT

; Output the text.
            xyouts, left, bottom-float(box_height/n_lines), all_text, $
              size=char_size, /NORMAL

          endif
          widget_control, event.top, /DESTROY
        end

    endcase

endif else if eventname EQ 'WIDGET_SLIDER' then begin

; Update the slider info.
    widget_control, info.char_sizer, get_value=char_size
    widget_control, info.char_size, set_value=char_size

endif

end

 ;************************************************************ *
;** Add_Text lets a user interactively add text to a graph **
 ;************************************************************ *

pro add_text, group_leader=group_leader

tlb = widget_base(title='Add Text', /COLUMN)
text_label = widget_label(tlb, value='Enter text below:', /ALIGN_LEFT)
text = cw_field(tlb, title=' ', value='', xsize=50, ysize=10)

char_sz_base = widget_base(tlb, /COLUMN, /FRAME, /ALIGN_CENTER)
```

```
char_size = cw_field(char_sz_base, title='Character Size:', xsize=2, $
                value=1, /NOEDIT)
char_sizer = widget_slider(char_sz_base, scroll=1, minimum=1, $
                    maximum=15, value=1, /SUPPRESS_VALUE, /DRAG)


exit_base = widget_base(tlb, /ROW, space=15)
ok_button = widget_button(exit_base, value=' OK ')
cancel_button = widget_button(exit_base, value=' Cancel ', /ALIGN_RIGHT)

widget_control, tlb, /REALIZE

info = {text:text,              $
       char_size:char_size,    $
       char_sizer:char_sizer   $
      }

widget_control, tlb, set_uvalue=info

xmanager, 'add_text', tlb, /MODAL, group_leader=group_leader

end

 ------------------------------------------------------------ -------------------
; Interactively choose a place to put a box (i.e. a legend box).
; [left, right, bottom, top] describe the edges of the box.
; By default, the box is dragged around the screen by the lower left corner,
; setting the UPPER_LEFT keyword drags it by the upper left corner.

pro place, left, right, bottom, top, upper_left=upper_left

; Save the current device setup, turn on XOR for "erase" feature.
device, get_graphics = prev_dvc, set_graphics = 6

x_size = !d.x_size
y_size = !d.y_size
x_vsize = !d.x_vsize
y_vsize = !d.y_vsize

; /NORMAL window coordinate ranges
xrange = [0.,1.]
yrange = [0.,1.]

if n_params() EQ 4 then begin

  Box_width = left > right - left < right
  Box_height = bottom > top  -  bottom < top
```

```
endif else begin

  Box_width = 0.2
  Box_height = Box_width * x_vsize/y_vsize

endelse

; Position the cursor.
; Note: !d.x_size is given in pixels.

x_curr = 0.5 * x_size          ; Horizontal center of graphics window.
y_curr = 0.5 * y_size          ; Vertical center.
tvcrs, x_curr, y_curr

width = x_vsize * Box_width
height = y_vsize * Box_height

if keyword_set(upper_left) then height = -height

; Set up box coordinates and draw it.

Box_x=[x_curr, x_curr+width, x_curr+width, x_curr, x_curr]
Box_y=[y_curr, y_curr, y_curr+height, y_curr+height, y_curr]
plots, Box_x, Box_y, /DEVICE

; Increment amounts

dx=1
dy=1

while 1 do begin

   x_old = x_curr
   y_old = y_curr

; Get latest cursor position
   cursor, x_curr, y_curr, /CHANGE, /DEVICE

   button_stat = !err

   if ( ((x_curr NE x_old) OR (y_curr NE y_old))  OR $
        (button_stat NE 0) ) then begin

; Erase the old box.
       plots, Box_x, Box_y, /DEVICE
       empty

; Keep the box in-bounds.
```

```
        xnew = x_curr > 0 < (x_size - width)
        ynew = y_curr > 0 < (y_size - height)

        if keyword_set(upper_left) then ynew = y_curr > abs(height) < y_size

        dx = xnew EQ x_curr
        dy = ynew EQ y_curr

        x_curr = xnew
        y_curr = ynew

; Draw the box in it's new position
        Box_x = [x_curr, x_curr+width, x_curr+width, x_curr, x_curr]
        Box_y = [y_curr, y_curr, y_curr+height, y_curr+height, y_curr]

        plots, Box_x, Box_y,/device
        empty

; Quit.
; If any mouse button was pressed, we quit.
; Note: for 2-button mice, remove the last condition (button_stat EQ 4).

        if (button_stat EQ 1) OR (button_stat EQ 2) or $
          (button_stat EQ 4) then begin
            plots, Box_x, Box_y, /DEVICE
            empty
            device, set_graphics = prev_dvc, cursor_standard=30

            left = poly(x_curr/x_vsize, xrange)
            right = left + Box_width
            bottom = poly(y_curr/y_vsize, yrange)
            top = bottom + Box_height

            return

        endif
    endif
endwhile

end
```

Subject: Re: Text processing on plots
Posted by davidf on Fri, 14 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Jeff Hicke <jhicke@al.noaa.gov> writes:

> I am interested in whether anyone has any IDL routines which allows one
> to overplot a variable length text string in the manner of a word
> processor. In other words, if I define a box (or width of a box), then
> the routine overplots the text within this box's width. If it is longer
> than one line, it will wrap to a new line. I'm looking at adding figure
> captions to IDL plots. Can anyone help?

You might have a look at my tip on calculating the width of a string
on my web page. (See also the program STR_SIZE while you are
there.) This doesn't do *exactly* what you want, but it may give
you some ideas. The STR_SIZE program calculates what character
size a string should be to fit into a defined portion of the display
window. I use it for captions on resizeable graphics windows.

Cheers!

David

David Fanning, Ph.D.
Fanning Software Consulting
2642 Bradbury Court, Fort Collins, CO 80521
Phone: 970-221-0438 Fax: 970-221-4762
E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com

Subject: Re: Text processing on plots
Posted by thompson on Sat, 15 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Jeff Hicke <jhicke@al.noaa.gov> writes:

> Hello,

> I am interested in whether anyone has any IDL routines which allows one
> to overplot a variable length text string in the manner of a word
> processor. In other words, if I define a box (or width of a box), then
> the routine overplots the text within this box's width. If it is longer
> than one line, it will wrap to a new line. I'm looking at adding figure
> captions to IDL plots. Can anyone help?

I think this does what you want.

Bill Thompson

==========

```
==================
 PRO WRITE_IN_BOX, X1, X2, Y1, Y2, TEXT, DATA=DATA, DEVICE=DEVICE, $
  NORMAL=NORMAL, MAXCHARSIZE=MAXCHARSIZE, COLOR=COLOR, $
  ALIGNMENT=ALIGNMENT
;+
; Project     : SOHO - CDS
;
; Name        : WRITE_IN_BOX
;
; Purpose     : Writes a text message within a box in a graphics window.
;
; Explanation : This procedure writes a short text message within a box-shaped
;  area in a graphics window.  The message may be split at word
;  boundaries into several lines, and the character size and
;  orientation may be adjusted for the text to fit within the box.
;
; Use         : WRITE_IN_BOX, X1, X2, Y1, Y2, TEXT
;
; Inputs      : X1, X2 = X coordinates of the box limits.
;  Y1, Y2 = Y coordinates of the box limits.
;  TEXT = ASCII text string containing the message.
;
; Opt. Inputs : None.
;
; Outputs     : None.
;
; Opt. Outputs: None.
;
; Keywords    : DATA     = If set, then the coordinates are in data units.
;         This is the default.
;  DEVICE     = If set, then the coordinates are in device units.
;  NORMAL     = If set, then the coordinates are in normalized
;         units.
;  MAXCHARSIZE = The maximum character size to use in displaying
;         the message.  If not passed, then determined from
;         !P.CHARSIZE.
;  COLOR      = Color to use to display the text.  The default is
;         !COLOR.
;  ALIGNMENT   = Controls the alignment of the text in the box.  A
;         value of 0 means to left justify the text, 0.5
;         means center it (default), and 1 stands for right
;         justification.
;
; Calls       : None.
;
; Common      : None.
;
; Restrictions: X2 must be greater than X1, and Y2 must be greater than Y1.
```

```
;
; Side effects: The appearance of the displayed message may not be optimal if
;  any words are separated by multiple blanks, or by tab
;  characters.
;
; Category    : Planning, Science.
;
; Prev. Hist. : None.
;
; Written     : William Thompson, GSFC, 7 July 1993.
;
; Modified    : Version 1, William Thompson, GSFC, 7 July 1993.
;  Version 2, William Thompson, GSFC, 24 September 1993.
;   Added ALIGNMENT keyword based on code provided by Jim
;   Pendleton, GRO/OSSE NU.
;  Version 3, William Thompson, GSFC, 21 September 1994
;   Added STRTRIM call.
;
; Version     : Version 3, 21 September 1994
;-
;
 ON_ERROR, 2
;
;  Check the number of parameters.
;
 IF N_PARAMS() NE 5 THEN MESSAGE, 'Syntax:  X1, X2, Y1, Y2, TEXT'
 IF (X1 GE X2) THEN MESSAGE,'X2 must be greater than X1'
 IF (Y1 GE Y2) THEN MESSAGE,'Y2 must be greater than Y1'
;
;  Select which alignment to use.  The default is 0.5 (centered text).
;
 IF N_ELEMENTS(ALIGNMENT) NE 1 THEN ALIGNMENT = 0.5
;
;  Convert the input parameters to device coordinates.
;
 IF KEYWORD_SET(NORMAL) THEN BEGIN
  DEV = CONVERT_COORD([X1,X2],[Y1,Y2],/NORMAL,/TO_DEVICE)
  XX1 = DEV(0,0)
  XX2 = DEV(0,1)
  YY1 = DEV(1,0)
  YY2 = DEV(1,1)
 END ELSE IF KEYWORD_SET(DEVICE) THEN BEGIN
  XX1 = X1
  XX2 = X2
  YY1 = Y1
  YY2 = Y2
 END ELSE BEGIN
  DEV = CONVERT_COORD([X1,X2],[Y1,Y2],/DATA,/TO_DEVICE)
```

```
  XX1 = DEV(0,0)
  XX2 = DEV(0,1)
  YY1 = DEV(1,0)
  YY2 = DEV(1,1)
 ENDELSE
;
;  Calculate the height and width of the box in characters.
;
 WIDTH  = (XX2 - XX1) / !D.X_CH_SIZE
 HEIGHT = (YY2 - YY1) / !D.Y_CH_SIZE
;
;  Decompose the message into words.
;
 WORDS = STR_SEP(STRTRIM(STRCOMPRESS(TEXT),2),' ')
;
;  Get the maximum character size.
;
 IF N_ELEMENTS(MAXCHARSIZE) EQ 0 THEN BEGIN
  IF !P.CHARSIZE GT 0 THEN MAXCHARSIZE = !P.CHARSIZE ELSE $
   MAXCHARSIZE = 1
 ENDIF
 IF MAXCHARSIZE LE 0 THEN MESSAGE,'MAXCHARSIZE must be positive'
;
;  Make two passes.  During the first pass, try fitting the text in
;  horizontally.  During the second pass, try vertically.
;
 BEST_SIZE = FLTARR(2)
 FOR I_PASS = 0, 1 DO BEGIN
;
;  Starting with the maximum character size, try to fit the text within the
;  window.  Format the text message into a series of lines, allowing each line
;  to grow until it can't fit within the box any more.
;
;
  CHARSIZE = MAXCHARSIZE
TRY_SIZE:
  LINES = STRARR(N_ELEMENTS(WORDS))
  I_LINE = 0
  FOR I_WORD = 0,N_ELEMENTS(WORDS)-1 DO BEGIN
   IF STRLEN(LINES(I_LINE)) EQ 0 THEN BEGIN
    LINES(I_LINE) = WORDS(I_WORD)
   END ELSE BEGIN
    TEST = LINES(I_LINE) + ' ' + WORDS(I_WORD)
    IF CHARSIZE*STRLEN(TEST) LE WIDTH THEN BEGIN
     LINES(I_LINE) = TEST
    END ELSE BEGIN
     I_LINE = I_LINE + 1
     LINES(I_LINE) = WORDS(I_WORD)
```

```
    ENDELSE
   ENDELSE
  ENDFOR
  LINES = LINES(0:I_LINE)
  N_LINES = N_ELEMENTS(LINES)
;
;  Test whether or not the text message will fit in the box.  If not, then
;  decrease the character size by 30% and try again.
;
  IF (CHARSIZE*MAX(STRLEN(LINES)) GT WIDTH) OR $
    (CHARSIZE*N_LINES GT HEIGHT) THEN BEGIN
   CHARSIZE = CHARSIZE * 0.7
   GOTO, TRY_SIZE
  ENDIF
;
;  Calculate how big the message can be and still fit within the box.  Allow a
;  little leeway for a margin.
;
  CHARSIZE = (WIDTH / (MAX(STRLEN(LINES)) + 1.)) < $
   (HEIGHT / (N_LINES + 1.)) < MAXCHARSIZE
;
;  Save the calculated character size, and the actual LINES array.  Then try it
;  rotated 90 degrees.
;
  BEST_SIZE(I_PASS) = CHARSIZE
  IF I_PASS EQ 0 THEN LINES_0 = LINES ELSE LINES_90 = LINES
  WIDTH  = (YY2 - YY1) / !D.X_CH_SIZE
  HEIGHT = (XX2 - XX1) / !D.Y_CH_SIZE
 ENDFOR
;
;  Get the color to use in displaying the message.
;
 IF N_ELEMENTS(COLOR) EQ 0 THEN COLOR = !COLOR
;
;  Choose which orientation to display the text in.  Give preference to the
;  horizontal direction.
;
 PREFERENCE = 0.3 * MAX(BEST_SIZE) / MAXCHARSIZE
 IF BEST_SIZE(1) GT (1+PREFERENCE)*BEST_SIZE(0) THEN BEGIN
  CHARSIZE = BEST_SIZE(1)
  ORIENTATION = 90
  LINES = LINES_90
 END ELSE BEGIN
  CHARSIZE = BEST_SIZE(0)
  ORIENTATION = 0
  LINES = LINES_0
 ENDELSE
 N_LINES = N_ELEMENTS(LINES)
```

```
;
;  Display the message.  Depending on the value of the ALIGNMENT keyword,
;  either left justify (ALIGNMENT=0), center (ALIGNMENT=0.5) or right justify
;  (ALIGNMENT=1) each line.
;
 FOR I_LINE = 0,N_LINES-1 DO BEGIN
  IF ORIENTATION EQ 90 THEN BEGIN
   XX = (I_LINE-0.5*N_LINES+0.75)*CHARSIZE*!D.Y_CH_SIZE +$
    0.5*(XX1 + XX2)
   YY = (1-ALIGNMENT)*YY1 + ALIGNMENT*YY2
  END ELSE BEGIN
   XX = (1-ALIGNMENT)*XX1 + ALIGNMENT*XX2
   YY = (0.5*N_LINES-I_LINE-0.75)*CHARSIZE*!D.Y_CH_SIZE +$
    0.5 * (YY1 + YY2)
  ENDELSE
  XYOUTS, XX, YY, LINES(I_LINE), /DEVICE, CHARSIZE=CHARSIZE, $
   ALIGNMENT=ALIGNMENT, COLOR=COLOR, FONT=-1, $
   ORIENTATION=ORIENTATION
 ENDFOR
;
 RETURN
 END
```