Subject: How to take proper tick labels with 'axis' command?
Posted by ICBM0926 on Wed, 07 Jan 2009 11:38:17 GMT
View Forum Message <> Reply to Message

Dear all
One may use 'axis' command to plot axes. However, one may need to use parameters like 'xticks' 'xminor' to acquire desired number of ticks.
The tick label may show improper number (ex. a decimal) with these parameters given. Can somebody tell me how to let IDL decide tick label properly and automatically while It can show desired number of ticks?

Best regards

Subject: Re: How to take proper tick labels with 'axis' command?
Posted by Jeremy Bailin on Thu, 08 Jan 2009 16:33:24 GMT
View Forum Message <> Reply to Message

On Jan 7, 6:38 am, ICBM0926 <ICBM0...@gmail.com> wrote:
> Dear all
> One may use 'axis' command to plot axes. However, one may need to use
> parameters like 'xticks' 'xminor' to acquire desired number of ticks.
> The tick label may show improper number (ex. a decimal) with these
> parameters given. Can somebody tell me how to let IDL decide tick
> label properly and automatically while It can show desired number of
> ticks?
>
> Best regards

XTICKFORMAT might be what you're looking for. You can feed it a function that translates the tick number into a string however you want.

-Jeremy.

Subject: Re: How to take proper tick labels with 'axis' command?
Posted by David Gell on Thu, 08 Jan 2009 19:08:35 GMT
View Forum Message <> Reply to Message

On Jan 7, 5:38 am, ICBM0926 <ICBM0...@gmail.com> wrote:
> Dear all
> One may use 'axis' command to plot axes. However, one may need to use
> parameters like 'xticks' 'xminor' to acquire desired number of ticks.
> The tick label may show improper number (ex. a decimal) with these
> parameters given. Can somebody tell me how to let IDL decide tick

> label properly and automatically while It can show desired number of
> ticks?
>
> Best regards

I've had this problem many many times. You need to set several
keywords:
xrange, xtickv, xminor, and xticks
for the x axis and the corresponding y and z keywords. I'm including
a
routine that I use to make neat time tick marks. It is rather
extensive because
the user can specify if the ticks will be on year, month, day ...
boundaries.

The basic method is to determine a range (xrange) that is an multiple
of the neat
tick mark locations, then specify the number of ticks (xticks),
remembering
that xticks is the number of intervals (there is one more tick mark
than the
value of xticks). You can quit here, or continue and specify values
for the
tick valuse (xtickv) and the number of minor ticks between the major
ticks
(xminor)

```
;+
; NAME:
;  inms_neat_ticks - determines values for tick labeling plot keywords
;
function inms_neat_ticks, anRange, $                ;; time range,
julian day
                        tickinterval=tickinterval, $ ;; unit of
ticks,
                        debug=debug
;
; ARGUMENTS:
;   anRange - a two element vector containing the range of Julian
Dates
;
; KEYWORDS:
;   tickinterval - specifies the tick interval to be used,
;                  choices are YEAR, MONTH, DAY, HOUR, QUARTER, MINUTE
;                  the value may be abreviated to 2 characters.
;   debug        - if set, diagnostic information is produced
;
```

```
; RETURN VALUE:
;   A structure containing the axis formatting values
;       {anRange: anAdjust, $   ;; a two element vector with the
;                               adjusted Julian Dates for the axis
;        anTickv: anTickv,  $   ;; an array tick mark locations
;
;        nTicks:  n_elements(anTickv)-1, $ ;; the number of tick
intervals
;        nMinor:  nMinor}   ;; the number of minor ticks between major
;
;   In the event of an error the scalar 0 is returned.
;
; EXAMPLE:
;
;    The following code fragment illustrates the use of this routine
and
;    inms_label_ticks.
;
; ;; convert times to Julian Dates
; ;; first PDS compiant times to UTC format
; ;; then UTC to Julian dates
; ;;
;     axOrdDate = inms_doy2utc(strupcase(axHkg.sclktime))
;     anJulDate = sprl_cvt_odate_jdate(axOrdDate.nDate,
axOrdDate.nMsecs)
;
; ;; determine axis formatting values
; ;;   Ticks are to be on the hour
; ;;
;     xTickDef = inms_neat_ticks([anJulDate[0], anJulDate[nPoints-1]],
$
;       tick='Hour')
;
; ;; specify the format for the tick labels
;     nDummy = inms_label_ticks(format='DOY')
;
; ;; create a plot
; ;;  using the fields in the xTickDef as values for the
; ;;  corresponding plot keyword
; ;;
;     plot, anJulDate, anY,$
;           position=anPlotPos, $
;           xrange=xTickDef.anRange, $  ;; obtained from
inms_neat_ticks
;           xtickv=xTickDef.anTickv, $  ;;
;           xminor=xTickDef.nMinor,  $  ;;
;           xticks=xTickDef.nTicks,  $  ;;
;           xtitle='!Ctime', $
```

```
;          xtickformat='inms_label_ticks', $
;          yrange=[nYmin, nYmax], $
;          ytitle=asLabels[anContIdx[0]], $
;          _extra=extra
;
;
; CHANGES:
;  2-Jun-2005 D. Gell Initial Coding
; 16-Jun-2005 D. Gell  Adjust tick mark location determination
; 17-Jun-2005 D. Gell  Correct adjustment of range for even day
;                      increments
; 09-Nov-2005 D. Gell  Added quarter hour major ticks
; 20-Sep-2006 DAG      Use error handler and inms_post_message
; 02-Jan-2006 DAG      Correct calculation of major ticks for quarter
hours
; 13-Mar-2008 DAG      Adjust tick intervals when units are minutes
; 22-Jul-2008 DAG      Adjust number of tick labels when units are
days
;-
 ;------------------------------------------------------------ ----------

   compile_opt logical_predicate, strictarr, hidden

   ;; establish error handler
   ;;
   bDebugSw = keyword_set(debug)
   nError = 0
   catch, nError
   if nError ne 0 then begin
      catch, /cancel
      inms_post_message, traceback=bDebugSw, console=inms_is_image()
or bDebugSW
      if bDebugSw then stop
      return,  0
   endif

   anMonLen = [31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]

   ;; validate arguments
   ;;
   if n_elements(anRange) ne 2 then begin
      message, 'You must supply a two element julian time range
vector'
   endif

   ;; decompose julian date into month, day, and year
   ;;
   anCalRange = sprl_cvt_jdate_mdy(anRange)
```

```
   anCalTimes = intarr(3, 2)
   anCalTimes[0, *] = anCalRange.nYear
   anCalTimes[1, *] = anCalRange.nMonth
   anCalTimes[2, *] = anCalRange.nDay

   case strupcase(strmid(tickinterval, 0, 2)) of

      'YE': begin
         ;; adjusted limits start at 1 Jan of first year
         ;; and extend to end of 31 Dec of last year
         anAdjust = julday([1, 12], [1, 31], $
            anCalTimes[0, *], [0, 23], [0, 59], [0, 59])
         nTickType = 1
         nQuanta = 365
      end

      'MO': begin
         ;; adjusted limits begin with 1 day of first month
         ;; and extend through last day of last month
         anAdjust = julday(anCalTimes[1, *], $
            [1, anMonLen[anCalTimes[1, 1]]], anCaltimes[0, *], $
            [0, 23], [0, 59], [0, 59])
         ntickType = 2
         nQuanta = 31
      end

      'DA': begin
         ;; adjusted limits begin at midnight of first day and
         ;; extend through last second of last day
         anAdjust = [ulong(anRange[0]), ulong(anRange[1]+1.0)] +
0.5D00
         nQuanta = 1.0
         nTickType = 3
      end

      'HO': begin
         ;; adjusted limits begin at begining of first hour and
         ;; extend through last second of last hour
         nQuanta = 1.0/2.4D01
         anAdjust = ulong(24*[anRange[0], anRange[1]+nQuanta/2.]) /
2.4D01
         nTicktype = 4
      end

      'QU': begin
         ;; adjusted limits begin on even quarter hour boundary
         ;; and extend through last quarter hour, 96 periods per
```

```
day
          nQuanta = 1.0/(9.6D01)
          anAdjust = ulong(96*[anRange[0], anRange[1]+nQuanta/2.]) /
9.6D01
          nTickType = 6
       end

       'MI': begin
          ;; adjusted limits begin at begining of first minute and
          ;; extend through last second of last minute
          nQuanta = 1.0/1.44d03
          anAdjust =  ulong(1440L*[anRange[0], $
             anRange[1]+nQuanta])/1.44D03
          ntickType = 5
       end



       else: begin
          message, 'Invalid tick "'+tickinterval+'" interval
specified'
       end
    endcase

    anAdjust = reform(anAdjust)
    nInterval = anRange[1]-anRange[0]
    nUnits = ceil(nInterval/nQuanta)

    ;; determine location of tick marks and
    ;; number of minor ticks on axis
    ;;
    case nTickType of
       6: begin
          ;; quarter hour units
          case 1 of
             nUnits le 4: begin
                anTickv = nQuanta * dindgen(nUnits+1)+anAdjust[0]
                nMinor = 0
             end
             nUnits le 16: begin
                nMinor = 4
                anTickv = 4 * nQuanta * dindgen(nUnits/4 + 1)$
                   + anAdjust[0]
             end
             else: begin
                nHours = nUnits/4
                case 1 of
                   nHours le 6: begin
                      anTickv = 4*nQuanta * dindgen(nHours+1) $
```

```
                    + anAdjust[0]
                  nMinor = 4
               end
               nHours le 12: begin
                  anTickv = 8 * nQuanta * dindgen(nHours/
2+1) $
                    + anAdjust[0]
                  nMinor = 4
               end
               else: begin
                  anTickv = 16 * nQuanta * dindgen(nHours/
4+1) $
                    + anAdjust[0]
                  nMinor = 4
               end
            endcase
         end
      endcase
   end
   5: begin
      ;; minute units
      ;;
      case 1 of
         nUnits le 5: begin
            anTickv = nQuanta * dindgen(nUnits+1) + anAdjust
[0]
            nMinor = 0
         end
         nUnits le 10: begin
            nMinor = 2
            anTickv = 2 * nQuanta * dindgen(nUnits/2 + 1)$
               + anAdjust[0]
         end
         nUnits le 30: begin
            nMinor = 5
            anTickv = 5 * nQuanta * dindgen(nUnits/5 + 1)$
               + anAdjust[0]
         end
         nUnits le 120: begin
            nMinor = 3
            anTickv = 15 * nQuanta * dindgen(nUnits/15 + 1) $
               + anAdjust[0]
         end
         else: begin
            nMinor = 4
            anTickv = 60 * nQuanta * dindgen(nUnits/60 + 1) $
               + anAdjust[0]
         end
```

```
      endcase
   end

   4: begin
      ;; hour units
      case 1 of
         nUnits le 6: begin
            anTickv = nQuanta * dindgen(nUnits+1) $
               + anAdjust[0]
            nMinor = 4
         end
         nUnits le 12: begin
            anTickv = 2 * nQuanta * dindgen(nUnits/2+1) $
               + anAdjust[0]
            nMinor = 4
         end
         nUnits le 48: begin
            anTickv = 8 * nQuanta * dindgen(nUnits/8 + 1) $
               + anAdjust[0]
            nMinor = 4
         end
         nUnits le 72: begin
            anTickv = 12 * nQuanta * dindgen(nUnits/12+1) $
               + anAdjust[0]
            nMinor = 6
         end
         else: begin
            anTickv = 24 * nQuanta * dindgen(nUnits/24+1) $
               + anAdjust[0]
            nMinor = 4
         end
      endcase
   end

   3: begin
      ;; day units
      ;;
      case 1 of
         nUnits le 5: begin
            anTickv = nQuanta * dindgen(nUnits+1) + anAdjust
[0]
            nMinor = 6
         end
         nUnits le 14: begin
            anTickv = 2 * nQuanta * dindgen(nUnits/2+1) $
               + anAdjust[0]
            nMinor = 4
         end
```

```
            nUnits le 35: begin
               anTickv = 7 * nQuanta * dindgen(nUnits/7+1) $
                  + anAdjust[0]
               nMinor = 7
            end
            nUnits le 70: begin
               anTickv = 14 * nQuanta * dindgen(nUnits/14+1) $
                  + anAdjust[0]
               nMinor = 7
            end
            nUnits le 280: begin
               anTickv = 28 * nQuanta * dindgen(nUnits/28+1) $
                  + anAdjust[0]
               nMinor = 4
            end
            else: begin
               anTickv = 56 * nQuanta * dindgen(nUnits/56+1) $
                  + anAdjust[0]
               nMinor = 4
            end
         endcase
      end

      2: begin
         ;; monthly tick marks

         case 1 of
            nUnits le 4:  anTickv = nQuanta * dindgen(nUnits+1)
            nUnits le 8:  anTickv = 2 * nQuanta * dindgen(nUnits/
2+1)
            nunits le 12: anTickv = 3 * nQuanta * dindgen(nUnits/
3+1)
            nunits le 24: anTickv = 4 * nQuanta * dindgen(nUnits/
4+1)
            nunits le 48: anTickv = 6 * nQuanta * dindgen(nUnits/
6+1)
            else:         anTickv = 12 * nQuanta * dindgen(nUnits/
12+1)
         endcase
         anTickv = anTickv + anAdjust[0]

         ;; convert JD to MDY, and make all day of months 1
         axTickMDY = sprl_cvt_jdate_mdy(anTickv+2)
         anTickv = julday(axTickMDY.nMonth, $
            1, axTickMDY.nYear, $
            0, 0, 0)
         nMinor = 0
      end
```

```
      1: begin
         ;; annual tick marks
         case 1 of
            nUnits le 2:  anTickv = nQuanta * dindgen(4*nUnits+5)/
4
            nUnits le 4:  antickv = nQuanta * dindgen(2*nUnits+3)/
2
            nUnits le 5:  antickv = nQuanta * dindgen(nUnits+2)
            nUnits le 10: antickv = 2*nQuanta * dindgen(nunits/
2+1)
            else:         antickv = 5*nQuanta * dindgen(nUnits/
5+1)
         endcase
         anTickv = anTickv + anAdjust[0]

         ;; convert JD to MDY, and make all day of months 1
         axTickMDY = sprl_cvt_jdate_mdy(anTickv)
         anTickv = julday(axTickMDY.nMonth, $
            1, axTickMDY.nYear, $
            0, 0, 0)
         nMinor = 0
      end
   endcase
   return, {anRange: anAdjust, $
      anTickv:anTickv, $
      nTicks: n_elements(anTickv)-1, $
      nMinor:nMinor}
end
```