IDL Week 3:
What we'll cover today

- All about plotting
- How IDL communicates with graphics devices
  - Screen vs. file
  - Colors
- 2D plots
  - Line
  - Scatter
  - Bar
- Contour plots
- Mapping
- Images

How IDL handles graphics

Direct Graphics in IDL use a device-oriented model. What this means is that you select a 'device' (usually screen or file), and IDL plots to that device. The default device is the screen ('X' or 'WIN', stored in *!D.NAME*). The other device you will most likely use is 'PS' (Postscript), although IDL supports other, less common formats.

Select your output device with the **set_plot** command:
set_plot, *device*

Note: Different devices have their own peculiarities, particularly when it comes to color.

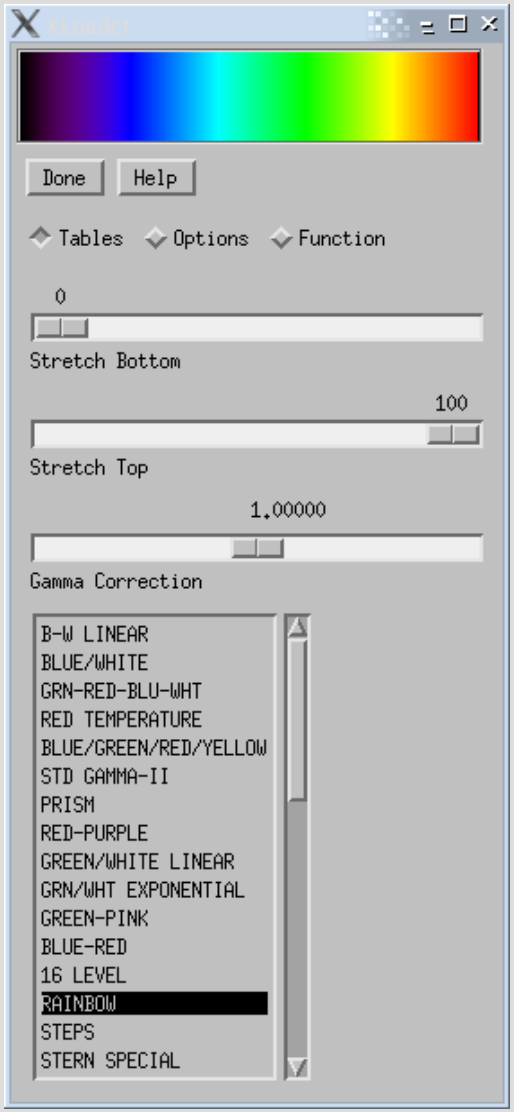
Colors in IDL

- There are 2 ways in which color is referenced in IDL.
  - Decomposed mode: treats every color as a 3-element vector representing the R,G,B components (so you have 16.7 million possible combinations)
  - Indexed mode: uses a color table that consists of 256 of these triplets.
- The display window uses decomposed mode by default, whereas PS can only used indexed mode.
- When specifying a color in decomposed mode, use a 3-element BYTARR
- When specifying a color in indexed mode, use a single number corresponding to the index.
- To change to indexed mode, use the command **device**,***true***=24,***decompose***=0,***retain***=2

Some Common Colors			
	R	G	B
White	255	255	255
Black	0	0	0
Red	255	0	0
Orange	255	128	0
Yellow	255	255	0
Green	0	255	0
Blue	0	0	255
Purple	82		104

Color Tables

- In 8-bit mode, colors are referenced by an index from 0-255 that represents their position in a color table
- IDL has several built-in color tables, to see what options you have, use the **loadct** or **xloadct** commands
- If you know the number of the color table you wish to use, load it by typing: **loadct**,*n*[,***/silent***]



Make Your Own Color Table

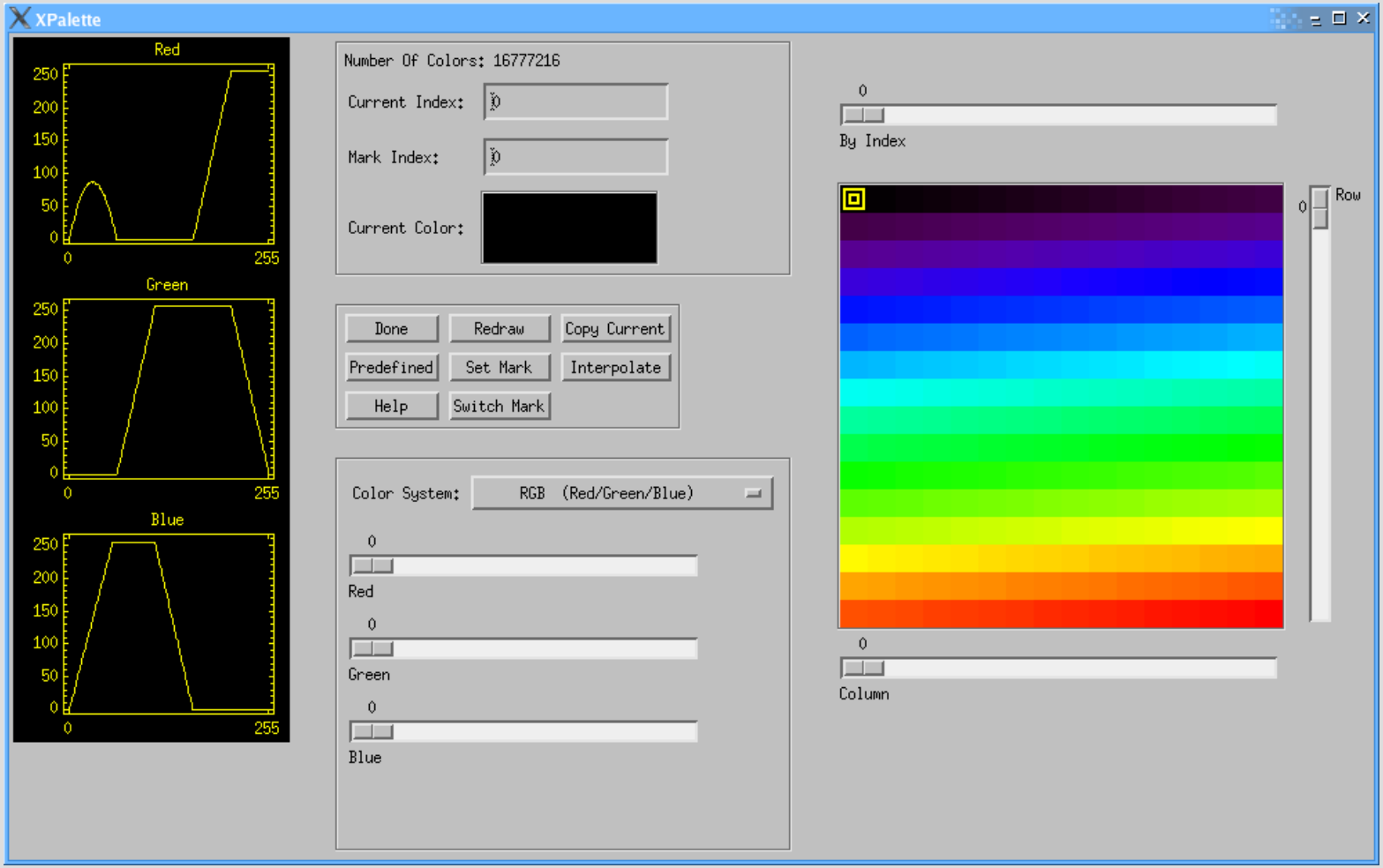
Option 1: Use TVLCT to load a custom color table from a 3 by Ncolor (up to 256) array:
TVLCT,*ct_array*[, ***/GET***] [, ***/HLS*** | , ***/HSV***]

Keywords:
/GET: puts current color table into ct_array
/HLS: Uses hue-saturation-lightness color system
/HSV: Uses hue-saturation-value color system

Note: in the HLS and HLV systems, H ranges from 0 to 360 while L and V range from 0 to 1.

Make Your Own Color Table

Option 2: Use **xpalette**:



Saving a custom color table

To save a color table for use in a future session of idl, use the **modifyct** command:

modifyct,*index*,*name*,*R*,*G*,*B*

Note: *index* may range from 0 to 255. Be careful not to replace an existing color table that you might want to use later! If *index* is greater than the lowest empty index, IDL will use the lowest empty index instead.

Example code (for a blue-white-red anomaly color table):

create_ct.pro

The PLOT command

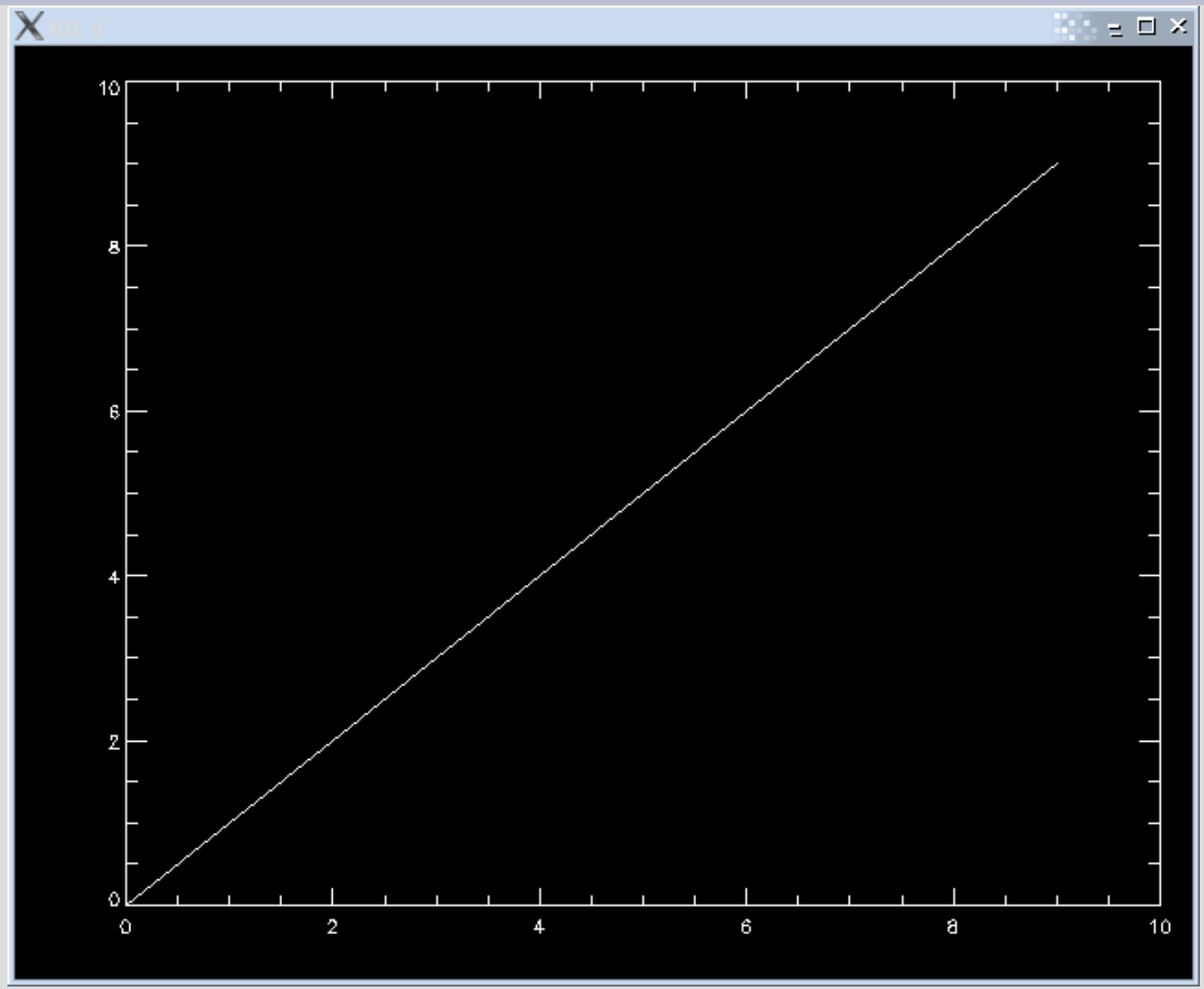
The plot command is extremely versatile. As a rule, it will try to do the best it can with as little information as you provide it, but you can provide it with a LOT of information:

PLOT, [*X*,] *Y* [, ***/ISOTROPIC***] [, ***MAX_VALUE***=*value*] [, ***MIN_VALUE***=*value*] [, ***NSUM***=*value*] [, /***POLAR***] [, ***THICK***=*value*] [, /***XLOG***] [, /***YLOG***] [, /***YNOZERO***]

Graphics Keywords: [, ***BACKGROUND***=*color_index*] [, ***CHARSIZE***=*value*] [, ***CHARTHICK***=*integer*] [, ***CLIP***=[*X0*, *Y0*, *X1*, *Y1*]] [, ***COLOR***=*value*] [, /***DATA*** | , /***DEVICE*** | , /***NORMAL***] [, ***FONT***=*integer*] [, ***LINESTYLE***={0 | 1 | 2 | 3 | 4 | 5}] [, /***NOCLIP***] [, /***NODATA***] [, /***NOERASE***] [, ***POSITION***=[*X0*, *Y0*, *X1*, *Y1*]] [, ***PSYM***=integer{0 to 10}] [, ***SUBTITLE***=*string*] [, ***SYMSIZE***=*value*] [, /***T3D***] [, ***THICK***=*value*] [, ***TICKLEN***=*value*] [, ***TITLE***=*string*]

[, {X | Y | Z}***CHARSIZE***=*value*]
[, {X | Y | Z}***GRIDSTYLE***=integer{0 to 5}]
[, {X | Y | Z}***MARGIN***=[*left*, *right*]]
[, {X | Y | Z}***MINOR***=*integer*]
[, {X | Y | Z}***RANGE***=[*min*, *max*]]
[, {X | Y | Z}***STYLE***=*value*]
[, {X | Y | Z}***THICK***=*value*]
[, {X | Y | Z}***TICK_GET***=*variable*]
[, {X | Y | Z}***TICKFORMAT***=*string*]
[, {X | Y | Z}***TICKINTERVAL***= *value*]
[, {X | Y | Z}***TICKLAYOUT***=*scalar*]
[, {X | Y | Z}***TICKLEN***=*value*]
[, {X | Y | Z}***TICKNAME***=*string_array*]
[, {X | Y | Z}***TICKS***=*integer*]
[, {X | Y | Z}***TICKUNITS***=*string*]
[, {X | Y | Z}***TICKV***=*array*]
[, {X | Y | Z}***TITLE***=*string*]
[, ***ZVALUE***=*value*{0 to 1}]

Example: Plot, findgen(10)



What does the plot command do?

If only a single array (*Y*) is specified, ***PLOT*** plots the values of *Y* vs. their index.

If 2 arrays (*X* and *Y*) are specified, ***PLOT*** will plot *Y* vs. *X*.

In addition to the plots themselves, ***PLOT*** creates axes and a data coordinate system which can be used by later commands.

IDL coordinate systems

When plotting in IDL, there are three coordinate systems that you may use: **data coordinates, device coordinates,** and **normal coordinates**.

The data coordinate system is established when you run the plot command or create a map coordinate system. Any command that requires position information will plot according to the data axes that have been established. If a data coordinate system has been established, most routines in IDL will assume you are using data coordinates by default.

Device coordinates are integers ranging from 0 to *!D.[XY}SIZE-1.* They may also be given in length units by using an appropriate keyword e.g., ***/inches***.

Normal coordinates simply express position as fraction of the plot window: [0,0] refers to the lower left corner and [1,1] refers to the upper right corner. Normal coordinates do not require any previous commands to be established, but you do have to specify the ***/NORMAL*** keyword for any command that takes both data and normal coordinates.

PLOT options - Axes

Often, you will want to change some aspect of the plot axes from the IDL defaults. These keywords allow manipulation of the axes:

/ISOTROPIC: identical scale for each axis

MAX_VALUE=*max*: don't plot values that exceed *max*

MIN_VALUE=*min*: don't plot values less than *min*

/XLOG and ***/YLOG***: logarithmic axes

/YNOZERO: don't extend Y axis to 0

[XY]RANGE = [*min*,*max*]: set the desired range

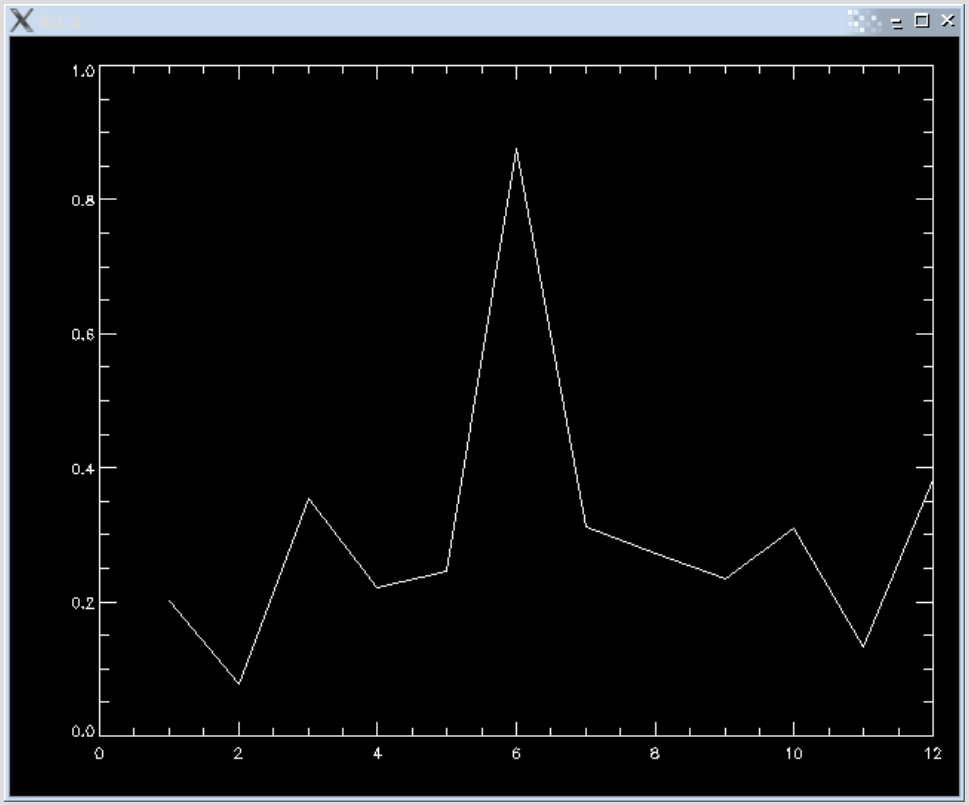
[XY]THICK = *thick*: set line thickness of axes

[XY]STYLE = *n*: set axis style

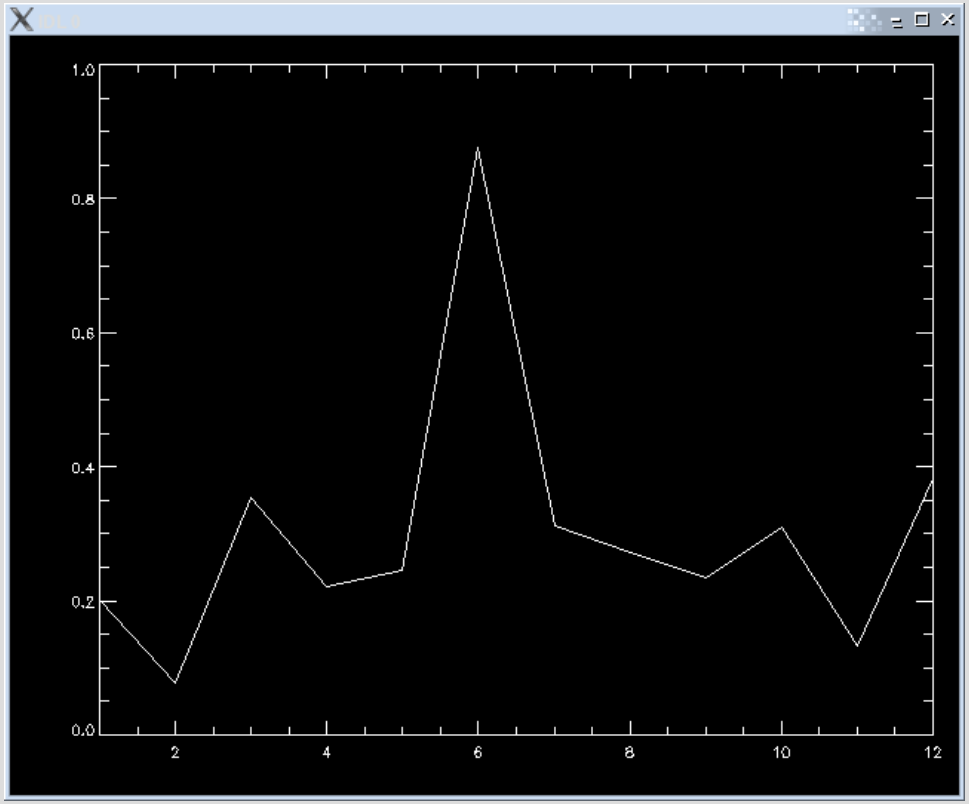
Style	Meaning
1	Force exact range
2	Extend axis range
4	Don't draw axis
8	One-sided axis
16	same as /YNOZERO

Example – Plot Axes

IDL> plot, month, rr_radar



plot, month, rr_radar, xrange=[1,12], xstyle=1



Plot options - Tickmarks

If you are not satisfied with IDL's default tick mark properties, they can be modified with the following keywords:

***[XY]MINOR**=n*: number of minor tick mark intervals

***[XY]TICKFORMAT**=fmt*: specify format code for tick labels

***[XY]TICKINTERVAL**=interval*: specify major (labelled) tick mark interval

***[XY]TICKLAYOUT**=n*: Specify tickmark style

Style	Meaning
0	Default
1	Only draw tick labels
2	Tick labels within boxes

***[XY]TICKLEN**=length*: specify length in normal units

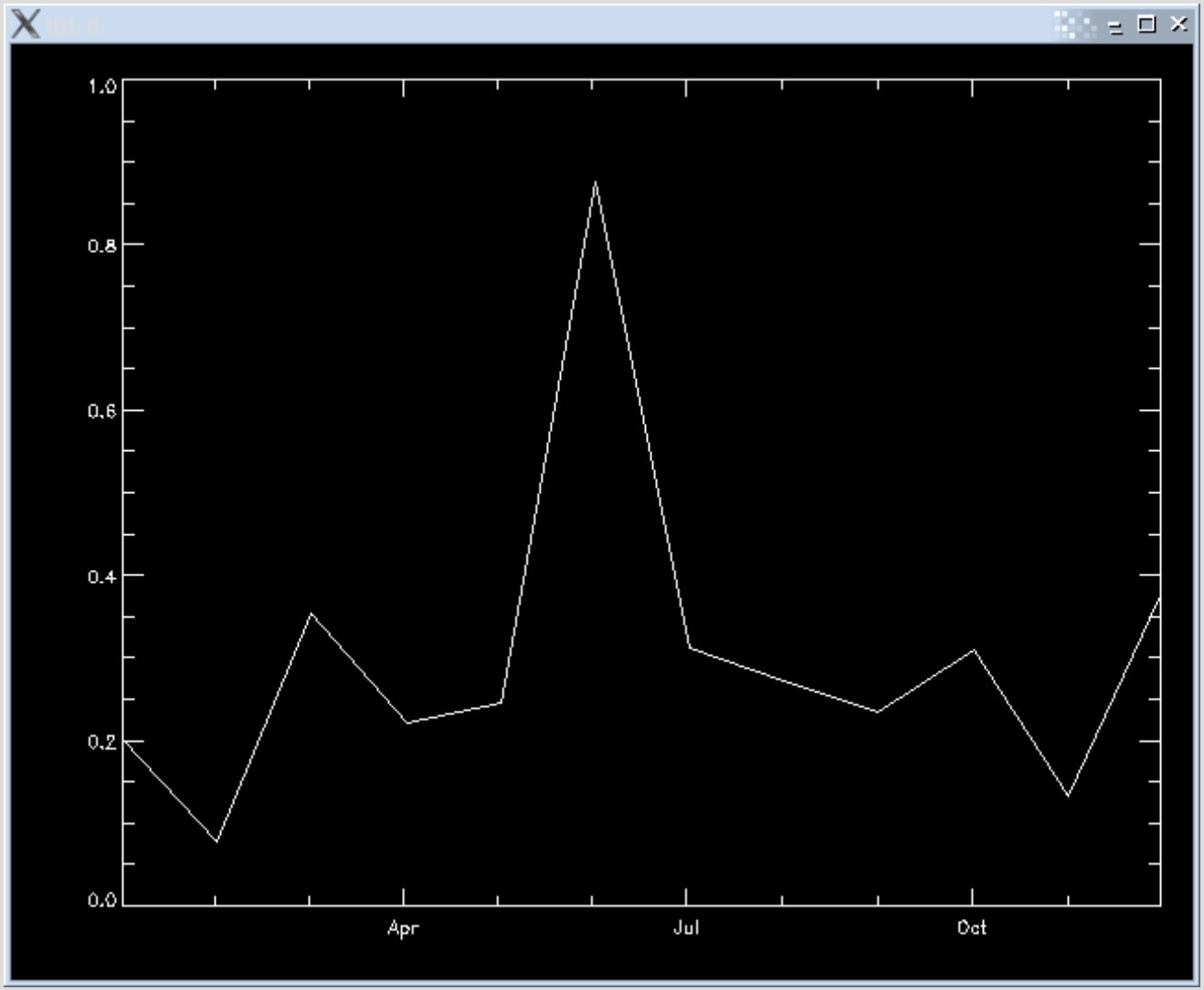
***[XY]TICKS**=n*: Number of major ticks to draw

***[XY]TICKUNITS**=type*: Useful for values that represent time (values must be in Julian days)

***[XY]TICKV**=array*: specify values for each tick mark

Example - Tickmarks

plot, (month-1)*365/12., rr_radar, xtickunits='month',xrange=[0,334],xstyle=1



Plot options - Titles

To label your axes, use the following keywords:
TITLE=*string*
XTITLE=*string*
YTITLE=*string*

To modify the font size, use the ***CHARSIZE***=*size* keyword (default is 1.0). By default, the main title will be 1.25 times the size of the X and Y titles. To change these separately, use ***[XY]CHARSIZE***.

For presentations and publications, it is a good idea to increase the font thickness for clarity. Use the ***CHARTHICK***=*thick* keyword for this purpose.

Fonts in IDL

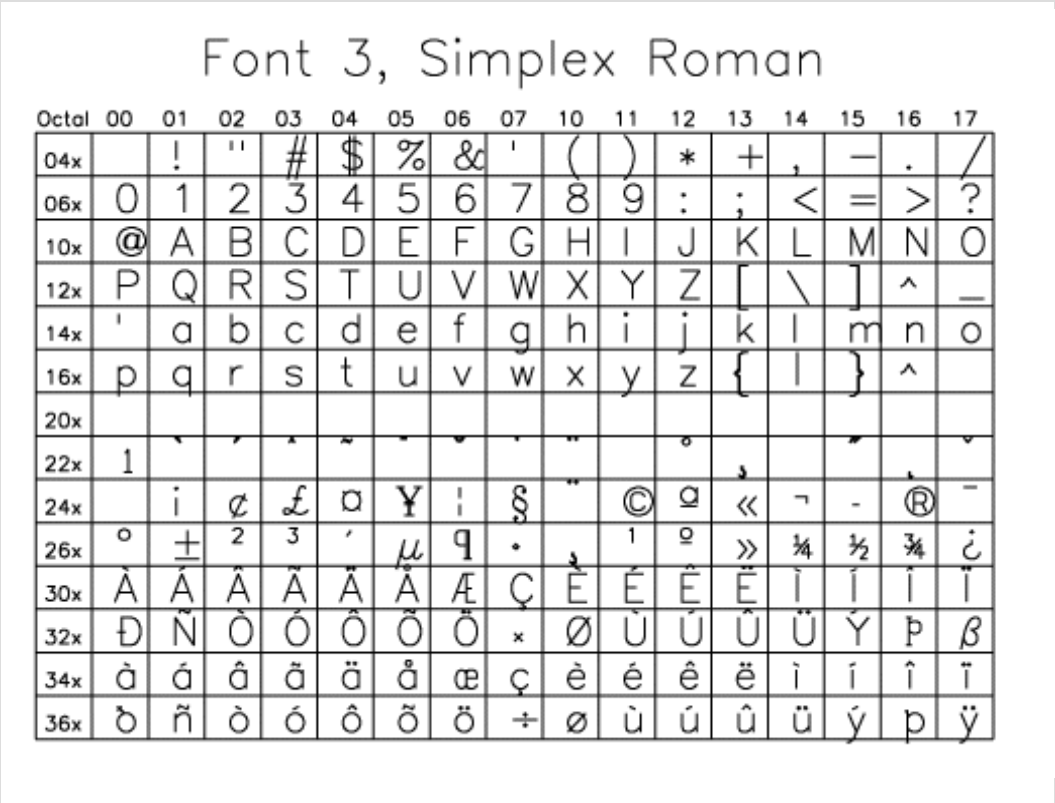
IDL can use three types of fonts: Hershey vector fonts, TrueType fonts, and device-specific fonts. The Hershey fonts are available to all devices and in every installation of IDL, and is the default font type (!P.FONT=-1)

To use a TrueType font, run the following command:

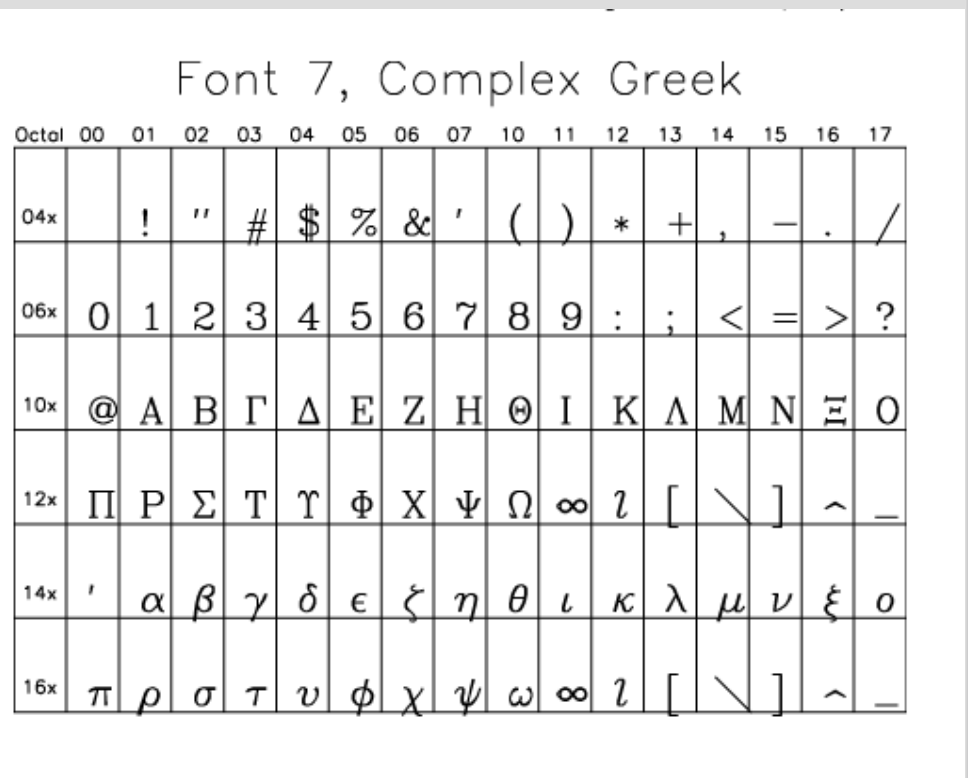
device,***set_font***=*name*,***/TT_FONT***

To use device fonts, set !P.FONT=0, followed by the **device**,***set_font***=*name* command.

Some Hershey Fonts



Octal	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17
04x		!	"	#	$	%	&	'	(	)	*	+	,	−	.	/
06x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
10x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
12x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
14x	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
16x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
20x																
22x																
24x		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
26x	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
30x	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
32x	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
34x	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
36x	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ



Octal	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17
04x		!	"	#	$	%	&	'	(	)	*	+	,	−	.	/
06x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
10x	@	A	B	Γ	Δ	E	Z	H	Θ	I	K	Λ	M	N	Ξ	O
12x	Π	P	Σ	T	Υ	Φ	X	Ψ	Ω	∞	ℓ	[	\	]	^	_
14x	'	α	β	γ	δ	ϵ	ζ	η	θ	ι	κ	λ	μ	ν	ξ	o
16x	π	ρ	σ	τ	υ	ϕ	χ	ψ	ω	∞	ℓ	[	\	]	^	_

To access a special character, use string(“nnnB)

Sub/Superscripts

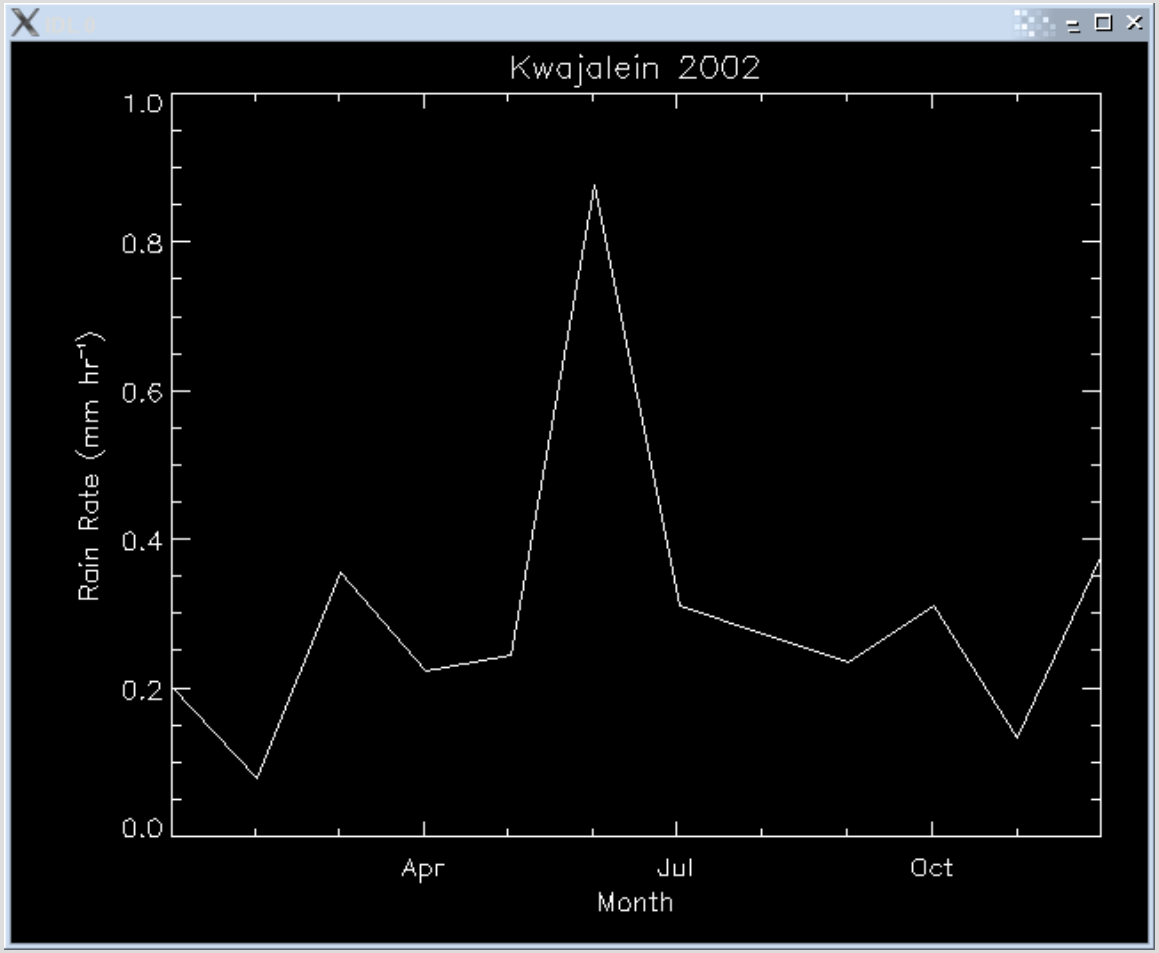
The use of the **!** character within a string allows you to embed various formatting commands.

To change font type, insert **!n**, where n is the Hershey font table (or TT/device equivalent, when available), followed by **!X** to revert to the initial font.

Subscript - **!I**
Superscript - **!E**
return to normal - **!N**
!! - display '!'

Example – Titles and Fonts

```
IDL> plot, (month-1)*365/12., rr_radar,
   xtickunits='month',xrange=[0,334],xstyle=1,ytitle='Rain Rate (mm hr!E-1!
   N)',charsize=1.5,title='Kwajalein 2002',xtitle='Month'
```



Plot options – colors, symbols, and styles

The background color and plot color can be specified with the ***BACKGROUND***=*c* and ***COLOR***=*c* keywords, respectively. Note that the axes will be also be plotted in the same color as the data! Use the ***/NODATA*** keyword to set background color and axes without plotting the data.

By default, IDL connects data point with a line, but these may also be represented by symbols. You can select a symbol with the ***PSYM***=*n* keyword (negative values connect the symbols with a line), and modify the size with ***SYMSIZE***=*size*.

The line itself may be modified with the ***LINESTYLE***=*n* and ***THICK***=*thickness* keywords.

PSYM	
1	Plus
2	Asterisk
3	Dot
4	Diamond
5	Triangle
6	Square
7	X

LINESTYLE	
0	Solid
1	Dotted
2	Dashed
3	Dash Dot
4	Dash Dot Dot
5	Long Dashes

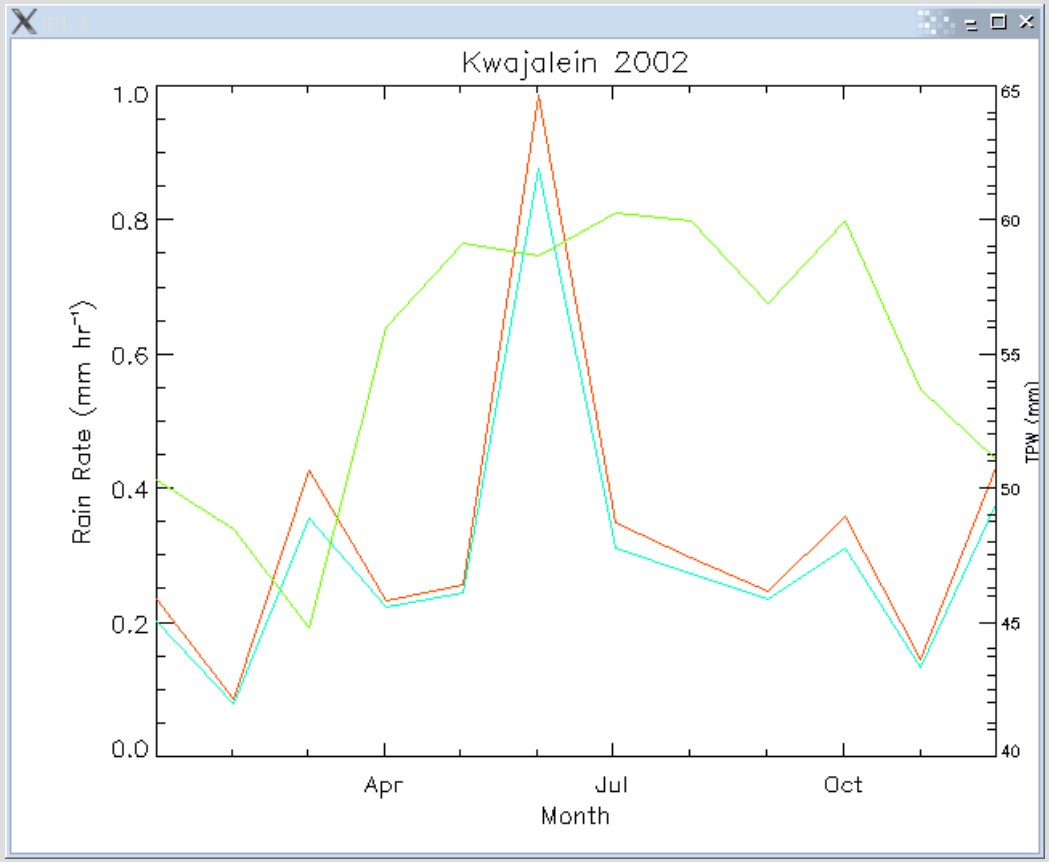
Multiple Datasets

To plot another dataset on the same axes, use the **oplot**,*x*,*y* command. Because the axes are not modified, **oplot** does not take as many keywords as plot, but the line color, symbol and style can all be modified.

To create a new axis, use the **AXIS** command:
AXIS,***YAXIS***=1,***/SAVE***,***YRANGE***=[*min*,*max*]

Example: Multiple Datasets

```
IDL> plot, (month-1)*365/12., rr_radar,
   xtickunits='month',xrange=[0,334],xst
   yle=1,ytitle='Rain Rate (mm hr!E-1!
   N)',charsize=1.5,title='Kwajalein
   2002',xtitle='Month',yrange=[0,1],/NO
   DATA,background=255,color=0
IDL> loadct,13,/silent
IDL> oplot, (month-1)*365/12., rr_radar,
   color=120
IDL> oplot, (month-1)*365/12.,
   rr_combined, color=240
IDL> axis,
   yaxis=1,/save,yrange=[40,65],color=0
   ,ytitle='TPW (mm)'
IDL> oplot, (month-1)*365/12., tpw,
   color=180
```



Adding text and lines: PlotS and XYOutS

You can add text anywhere you want on the plot using the XYOutS command:

XYOutS,*X0*,*Y0*,*string[*,***charsize***=*value][*, ***charthick***=*value]*[,***/NORMAL***]

You can plot a line using the PlotS command:

PlotS, *xvalues*, *yvalues*[,***color***=*c*][,***linestyle***=*n*] [***thick***=*value*][,***psym***=*n*]

Adding a legend

You could use a combination of **XYOutS** and **PlotS** commands to create a legend, but fortunately, someone has already done this for you. The legend procedure (legend.pro) uses the following syntax:

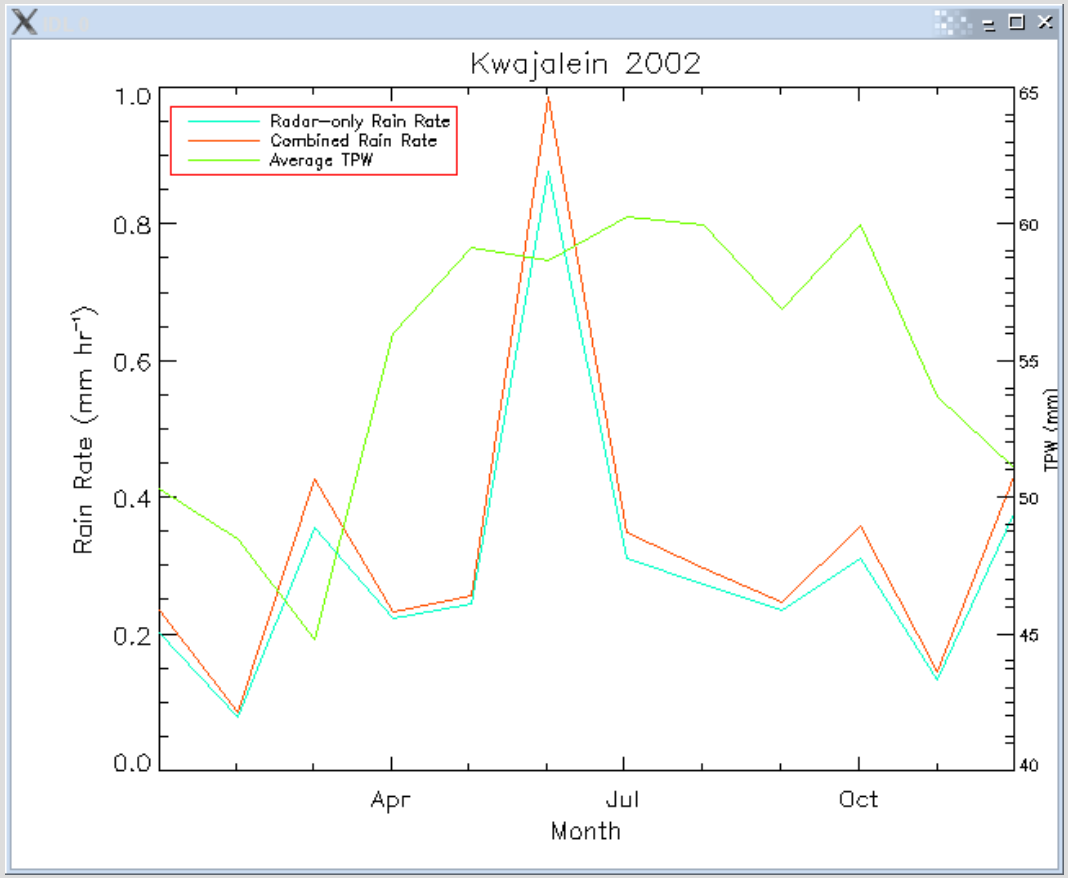
legend,*items*[,***linestyle***=*array*][,***psym***=*array*][,***colors***=*array*][,***/right***|***/left***][,***/top***|***/bottom***]

items is an array of strings that you wish to be displayed next to their corresponding symbol/line type, while *linestyle* and *psym* are arrays of their respective values.

(more options available, see legend.pro for details)

Example: Legend

```
IDL> plot, (month-1)*365/12., rr_radar,
   xtickunits='month',xrange=[0,334],xst
   yle=1,ytitle='Rain Rate (mm hr!E-1!
   N)',charsize=1.5,title='Kwajalein
   2002',xtitle='Month',yrange=[0,1],/NO
   DATA,background=255,color=0
IDL> loadct,13,/silent
IDL> oplot, (month-1)*365/12., rr_radar,
   color=120
IDL> oplot, (month-1)*365/12.,
   rr_combined, color=240
IDL> axis,
   yaxis=1,/save,yrange=[40,65],color=0
   ,ytitle='TPW (mm)'
IDL> oplot, (month-1)*365/12., tpw,
   color=180
IDL> legend,['Radar-only Rain
   Rate','Combined Rain Rate','Average
   TPW'],linestyle=0,colors=[120,240,18
   0],textcolors=0
```



Positioning and Multiple Plots on a Page

The ***POSITION*** keyword allows you to specify (in device or normal coordinates) the lower-left and upper-right corners of the x and y axes: ***POSITION***=[*x0*,*y0*,*x1*,*y1*].

The ***/NOERASE*** keyword instructs IDL to not erase the screen when drawing a plot, so a combination of ***POSITION*** and ***/NOERASE*** can be used to create multiple plots on a device.

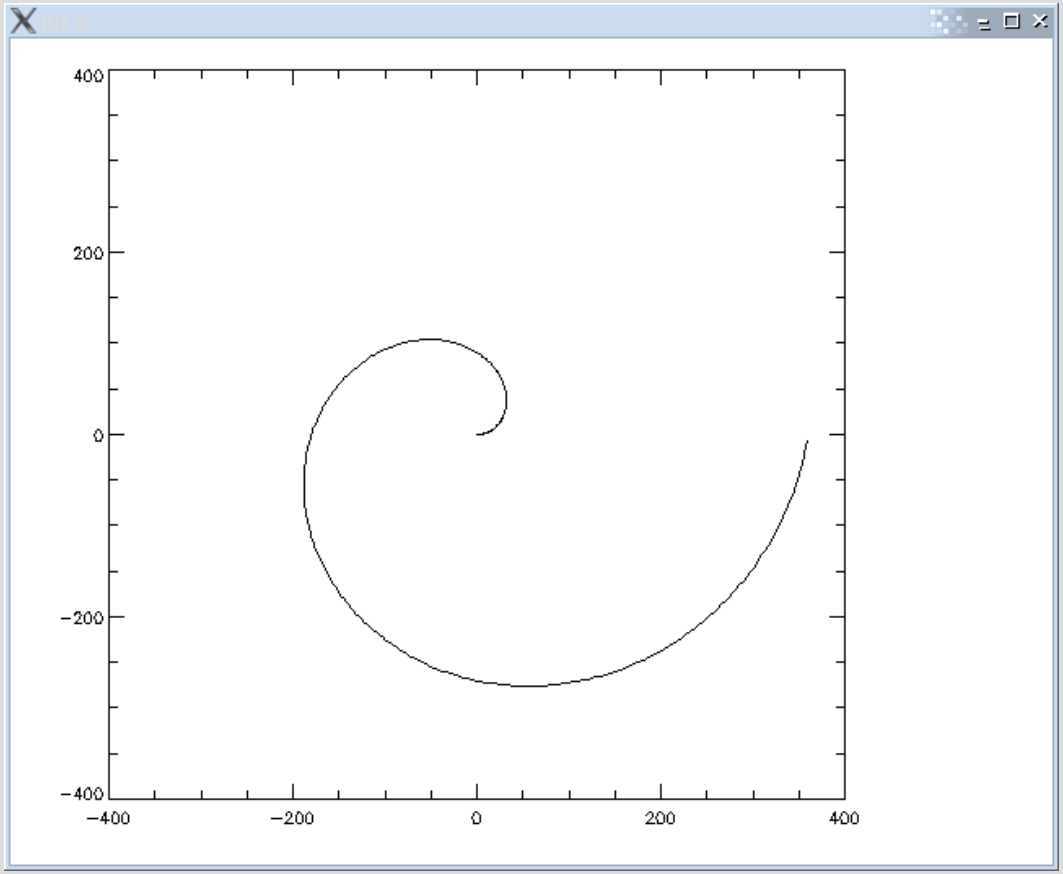
However, there is an easier way to create evenly-spaced plots, using the *!p.multi* system variable. *!p.multi* is a 5-element integer array:

!p.multi[0]: number of plots remaining (set to 0 for a new page)
!p.multi[1]: number of columns per page
!p.multi[2]: number of rows per page
!p.multi[3]: number of plots in the z direction (need a 3D coordinate system)
!p.multi[4]: =0 to fill rows then columns or 1 for columns then rows

Polar Plots

When the /POLAR keyword is specified, the x-values are interpreted as radius, and the y-values are interpreted as angle (theta).

```
IDL> plot, findgen(360), !DTOR*findgen(360),/POLAR,xrange=[-400,400],yrange=[-400,400],/isotropic,background=255,color=0
```



Bar Plots

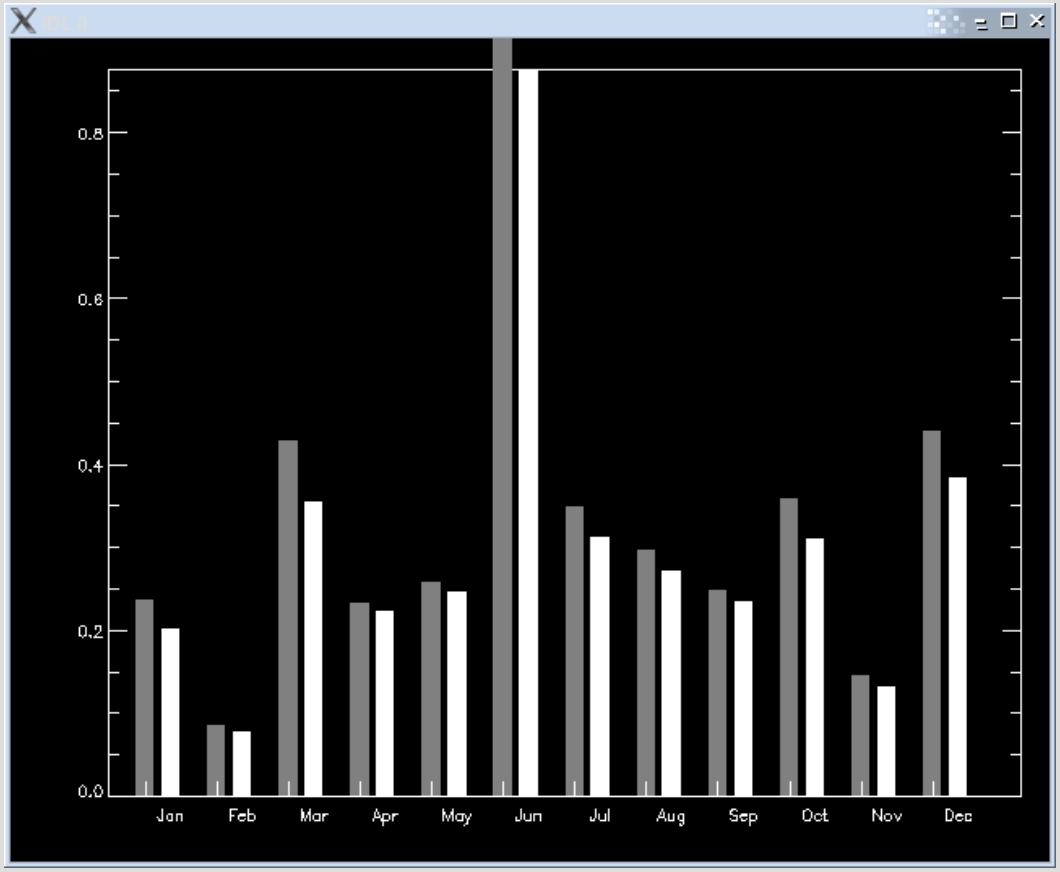
Instead of creating a data coordinate system followed by lots of calls to **PlotS**, the **BAR_PLOT** procedure can be used to create bar charts.

BAR_PLOT, *Values* [, ***BACKGROUND****=color_index*] [, ***BARNAMES****=string_array*] [, ***BAROFFSET****=scalar*] [, ***BARSPACE****=scalar*] [, ***BARWIDTH****=value*] [, ***BASELINES****=vector*] [, ***BASERANGE****=scalar*{0.0 to 1.0}] [, ***COLORS****=vector*] [, /***OUTLINE***] [, /***OVERPLOT***] [, /***ROTATE***] [, ***TITLE****=string*] [, ***XTITLE****=string*] [, ***YTITLE****=string*]

Multiple sets of bars can be created using the ***BARSPACE*** and ***/OVERPLOT*** keywords.

Bar plot example

```
IDL>
   bar_plot,rr_radar,background=0,barnames=['Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','
   Oct','Nov','Dec'],colors=bytarr(12)+255,barspace=0.7
IDL> bar_plot,rr_combined,colors=bytarr(12)+128,/overplot,barspace=0.7,baroffset=1.5
```



Contour Plots

The **CONTOUR** command is used to create contour plots. As with **PLOT**, there are several options:

CONTOUR, *Z* [, *X*, *Y*] [, ***C_ANNOTATION***=*vector_of_strings*] [, ***C_CHARSIZE***=*value*] [, ***C_CHARTHICK***=*integer*] [, ***C_COLORS***=*vector*] [, ***C_LABELS***=*vector*{each element 0 or 1}] [, ***C_LINESTYLE***=*vector*] [{, ***/CELL_FILL*** | , ***/FILL***} | [, ***C_ORIENTATION***=*degrees*] [, ***C_SPACING***=*value*]] [, ***C_THICK***=*vector*] [, ***/CLOSED***] [, ***/DOWNHILL***] [, ***/FOLLOW***] [, ***/IRREGULAR***] [, ***/ISOTROPIC***] [, ***LEVELS***=*vector*] [, ***NLEVELS***=*integer*{1 to 60}] [, ***MAX_VALUE***=*value*] [, ***MIN_VALUE***=*value*] [, ***/OVERPLOT***] [{, ***/PATH_DATA_COORDS***, ***PATH_FILENAME***=*string*, ***PATH_INFO***=*variable*, ***PATH_XY***=variable} | , ***TRIANGULATION***=*variable*] [, ***/PATH_DOUBLE***] [, ***/XLOG***] [, ***/YLOG***] [, ***ZAXIS***={0 | 1 | 2 | 3 | 4}]

In addition, **CONTOUR** accepts most of the graphics keywords that **PLOT** does (***position***, *title*, ***/isotropic***, etc.).

Contour: Data Orientation

If your data is already in a 2D array spaced at regular intervals, the contours can be plotted with:

contour, *Z*, *X*, *Y*

where *X* and *Y* are 1D arrays of the x and y coordinates, and *Z* is a 2D array of size nx by ny.

If the data is not regularly spaced, then *Z*, *X*, and *Y* must all be 1D arrays where *X* and *Y* correspond to *Z* at the same position in the array. You must also include the ***/IRREGULAR*** keyword:

contour, *Z*, *X*, *Y*, ***/IRREGULAR***

Note that contouring an irregular array when a mapping projection has been enabled will first call a triangulation procedure that modifies the contents (and dimensions) of *Z*, *X*, and *Y*. Thus, they must be whole arrays (not subsets), and if a different dataset *Z* is to be plotted later, the original *X* and *Y* should be retained in a separate variable.

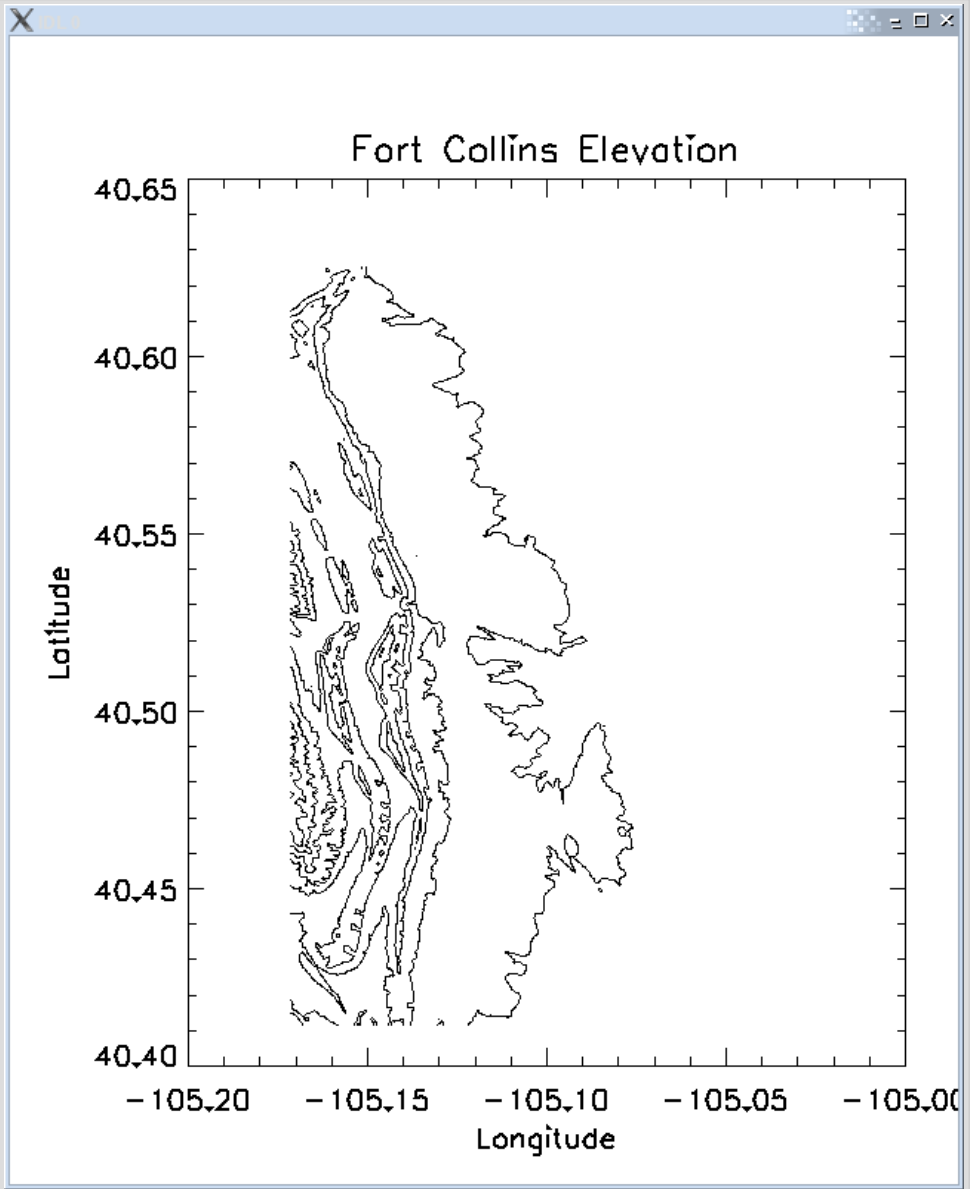
Contour Levels

To specify the number of contour levels, use the ***NLEVELS***=*n* keyword, and IDL will create evenly-spaced intervals based on your data range (or ***MIN_VALUE*** and ***MAX_VALUE***)

To specify specific contour levels, use the ***LEVELS***=*array* keyword.

Example:

```
IDL> contour, dem, lon, lat,
   background=255, color=0, levels =
   1500+50*(findgen(10)+1),
   title='Fort Collins Elevation',
   xtitle='Longitude',
   ytitle='Latitude',/isotropic
```



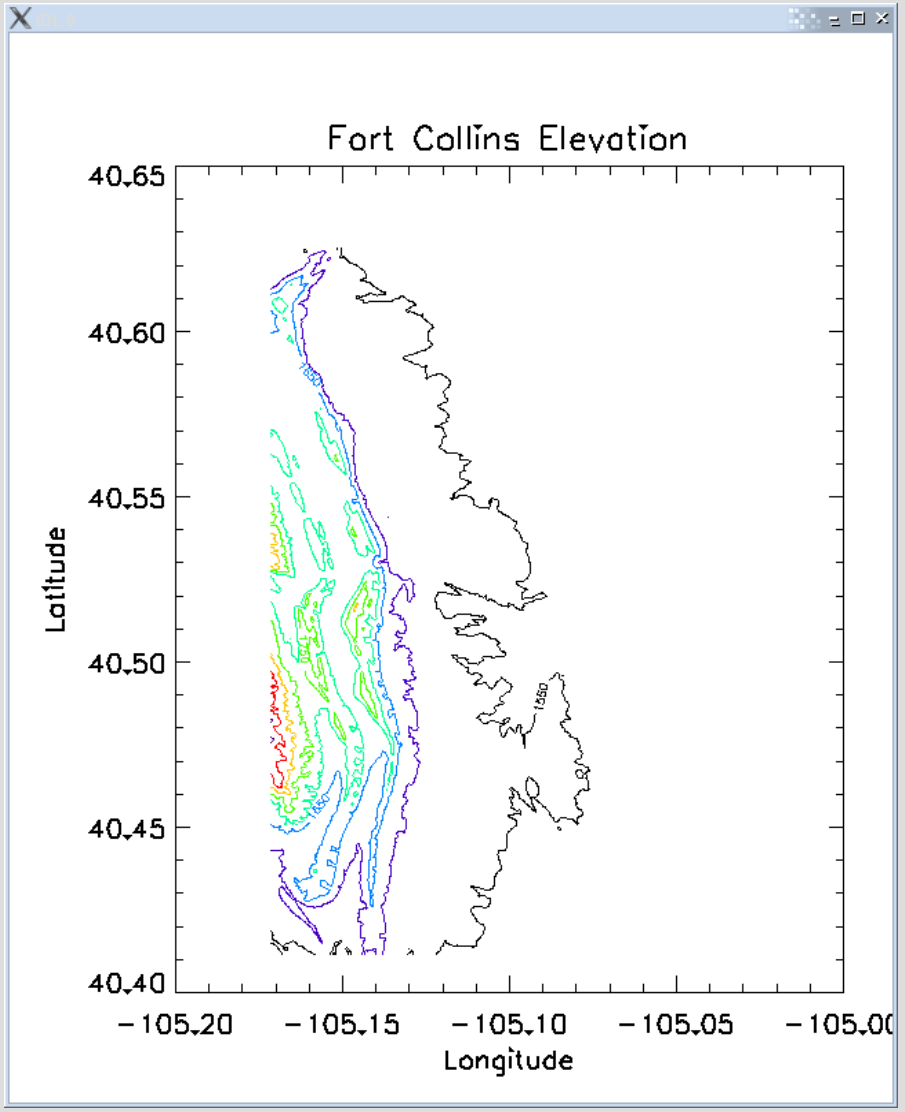
Contour Labels

IDL will label every other contour if the ***/FOLLOW*** keyword is used. For more control over labeling, you can can use ***C_LABELS***=*array*, where the levels corresponding to a nonzero value are labeled. To specify the text of each label, use the ***C_ANNOTATION***=*string_array* keyword.

The linestyle, color, and thickness of each level can be specified with the ***C_LINESTYLE, C_COLORS***, and ***C_THICK***, respectively.

Example:

```
IDL> contour, dem, lon, lat, background=255,
   color=0, levels = 1500+50*(findgen(10)+1),
   title='Fort Collins Elevation', xtitle='Longitude',
   ytitle='Latitude',/isotropic,charsize=2,charthick=2,/
   follow,c_colors=40*indgen(10)
```



Filled Contours

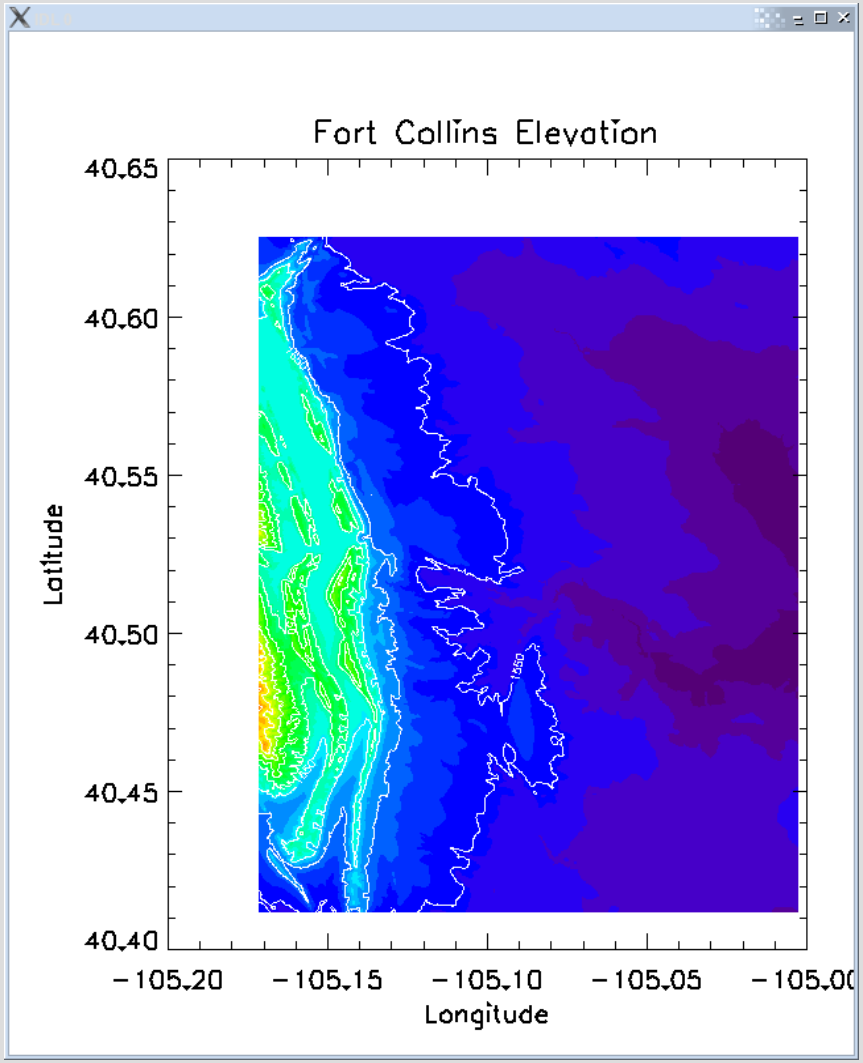
The ***/FILL*** keyword will fill in contours with evenly-spaced colors from the current color table, or the colors in ***C_COLOR***, if specified.

When using a map projection, or if there is missing data, use ***/CELL_FILL*** instead.

To label filled contours, call **CONTOUR** again with the ***/OVERPLOT*** keyword.

Example:

```
IDL> contour, dem, lon, lat, background=255,
   color=0, nlevels=32, /fill, title='Fort Collins
   Elevation',xtitle='Longitude',ytitle='Latitude',/isotro
   pic,charsize=2,charthick=2
IDL> contour, dem, lon, lat, levels =
   1500+50*(findgen(10)+1), /follow, /overplot
```



Colorbars!

IDL does not come with a built-in colorbar routine, but someone else has already done the work for you. The **COLORBAR** procedure supplied in colorbar.pro utilizes the following keywords:

VMIN=*vmn* Minimum value of color bar parameter (def=0).
VMAX=*vmx* Maximum value of color bar parameter (def=top).
CMIN=*cmn* Color that represents *vmn* (def=0).
CMAX=*cmx* Color that represents *vmx* (def=top).
CTICKNAME=*ctn* Array of color bar axis labels.
/HORIZONTAL Colors vary horizontally (def).
/VERTICAL Colors vary vertical.
/BOTTOM Horizontal axis on bottom (def).
/TOP Horizontal axis on top.
/RIGHT Vertical axis on right (def).
/LEFT Vertical axis on left.
Plus all keywords accepted by **PLOT**.

Maps

Often, our data will be geographic in scope, and we will want to plot a map for reference. IDL has a lot of flexibility in this regard, but the basic process is fairly simple:

1. Set a map projection and region (this defines the data coordinates)
2. Plot or contour your data (using OPLOT or /OVERPLOT)
3. (Optionally) Draw coast/country outlines
4. (Optionally) Draw a lat/lon grid

Setting the Map Projection

The **MAP_SET** procedure can be used with 18 projections (type ?MAP_SET for a list). For most common projections, the most often-used keywords are ***LIMIT***, and ***/ADVANCE***.

MAP_SET,*lat0*,*lon0*: creates a map spanning the globe centered on lat0,lon0 in the default (cylindrical) projection

MAP_SET,*lat0*,*lon0*,***LIMIT***=[*minlat*,*minlon*,*maxlat*,*maxlon*]: creates a projection over the region specified by ***LIMIT***

MAP_SET,*lat0*,*lon0*,***/ADVANCE***: Moves on to the next available plotting space when used in conjunction with !p.multi

Plotting on a map projection

Now that the data coordinates have been established, point or vector data can be plotted with **OPLOT**, and continuous data can be contoured with **CONTOUR** and ***/OVERPLOT*** (remember to use ***/CELL_FILL*** for filled contours and whole arrays with ***/IRREGULAR***)

To plot coastlines, rivers, and political boundaries, use the **MAP_CONTINENTS** procedure (use ***/HIRES*** for regions smaller than about 5x5 degrees):

MAP_CONTINENTS [, ***/COASTS***] [, ***/CONTINENTS***] [, ***/COUNTRIES***] [,***FILL_CONTINENTS***={1 | 2}[, ***/HIRES***] [, ***/RIVERS***][, ***/USA***]

To plot a lat/lon grid, use **MAP_GRID**:

MAP_GRID,***LATDEL***=*dy*,***LONDEL***=*dx*

Example Map

See plot_sst.pro

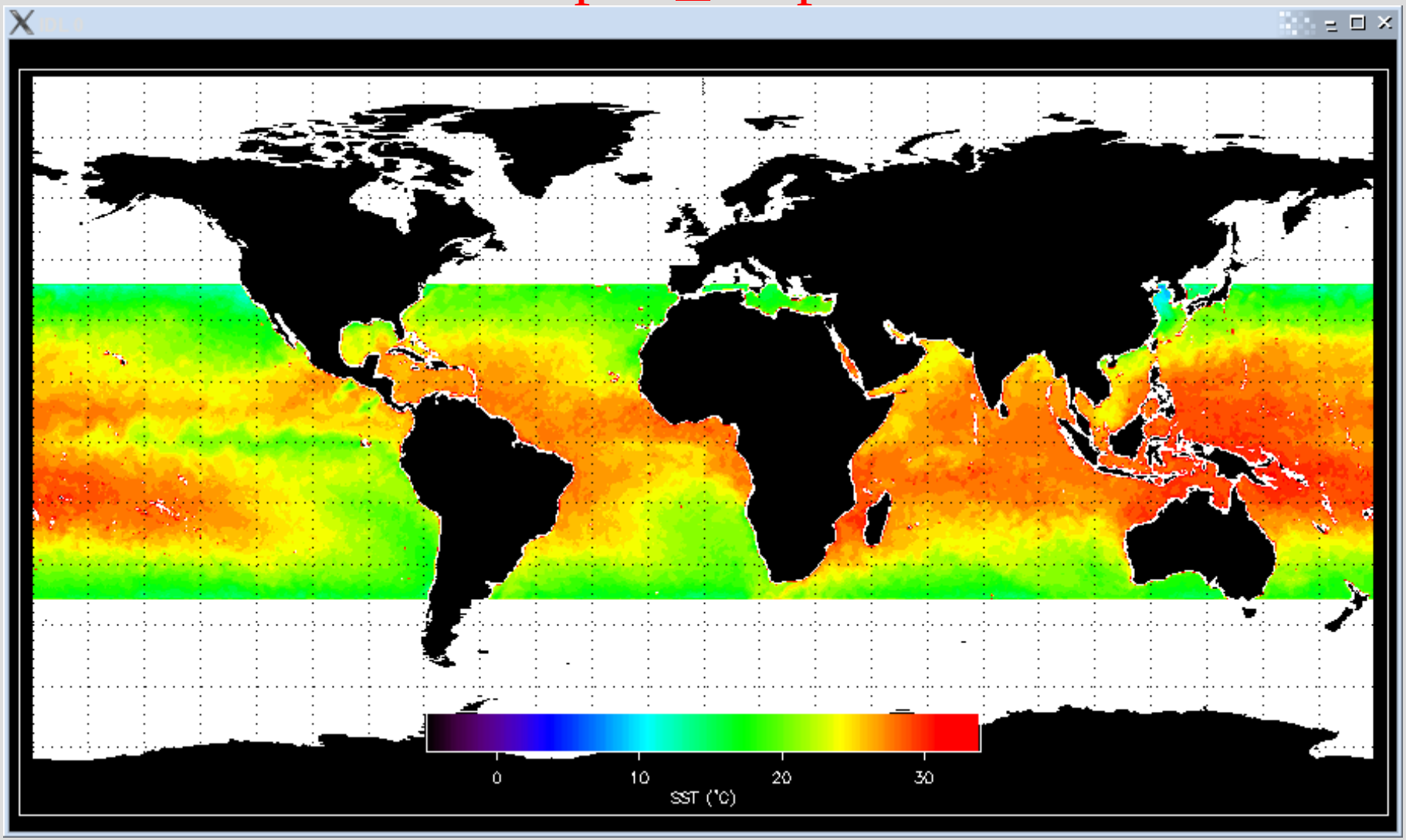


Image Display

If you ran the plot_sst.pro program, you may have noticed that the /cell_fill keyword is rather slow. For regularly spaced data that is very dense and/or noisy, treating it as image data is another option.

TV plots a 2D array to the screen as a bitmap (where each element is converted to a value from 0 to 255 to obtain a color from the color table):

TV,*array*,*x0*,*y0*,***xsize***=*dx*,***ysize***=*dy*

To scale the image so that the minimum value becomes 0 and the maximum becomes 255, use **TVscl**.

MAP_IMAGE

The TV command does not have an option to use existing data coordinates, so you need to use **MAP_IMAGE** to warp the pixels to the projection defined by **MAP_SET**:

result = **MAP_IMAGE**(*image*,*Startx*,*Starty*,
LATMIN=*latmin*, ***LONMIN***=*lonmin*,
LATMAX=*latmax*,
LONMAX=*lonmax*,***xsize***=*xs*,***ysize***=*ys*)
TV, *result*, *Startx*, *Starty*,***xsize***=*xs*,***ysize***=*ys*

Creating Postscript files

To create publication-quality plots, you will want to use the 'PS' device, which writes output in the Postscript format. All plotting functions work as normal, however, there are a few additional lines of code you will have to include:

set_plot, 'PS'
device, ***color***=1, ***filename***=*string.[e]ps*
[, ***/encapsulated***], ***xsize***=*xs*, ***ysize***=*ys*, [***/Inches***],
BITS_PER_PIXEL=8

...plotting commands...

device,*/close*