

Basic File I/O and Plotting with IDL

A.K.A: At Home With IDL

John Rausch
10 December 2009

Liberally taken from a CSU IDL course

Quick & Dirty Ways to Read ASCII Data

- The `READ_ASCII` function: Places freely formatted data into a structure
- The `ASCII_TEMPLATE` function: Provides a *relatively* easy to use graphical interface

Opening Files

- All basic input/output in IDL is done with logical unit numbers.
- A logical unit number is a pipe or conduit that is connected between IDL and the data file that you want to read from or write to.
- There are three types of open commands
 - `open r` - Open a file for Reading
 - `open w` - Open a file for Writing
 - `open u` - Open a file for Updating

Opening Files

- The syntax of these three commands is the same: name of the command, logical unit number (lun), file
`openw, 20, 'filename.txt'`
- You can use logical a unit numbers directly (and manage them yourself) ... as in the above example.
- You can also have IDL manage logical unit numbers by using `/GET_LUN`
`openr, lun, 'filename.txt', /GET_LUN`
- When you are finished with the logical unit numbers you can:
 - close it using the `close` command (in the first example) - `close, 20`
 - or free the lun using `FREE_LUN - FREE_LUN, lun`

Reading/Writing Formatted Data

- IDL distinguishes between two types of formatted files with regard to reading and writing data:
 - Free File Format: uses either commas or whitespace (tabs and spaces) to distinguish each element in the file. It is more informal than an explicit file format.
 - An explicit format file is formatted according to rules specified in a format statement. The IDL format statement is similar to format statements you might use in FORTRAN.

Always make sure...

- When you are done with a file, make sure that you close it! To do this type:

```
IDL> FREE_LUN, 1un
```

Explicit File Format

- To read/write an explicit file format, use the same `readf` and `printf` commands that were used for free format files, except now the file format is explicitly stated with the `format` keyword.

Format Specifier	What do they do
I	Specifies integer data
F	Specifies floating point data
D	Specifies double precision data
E	Specifies floating point data using scientific notation
A	Specifies character data
nx	Skip n character spaces

Simple Graphical Displays

- The quickest way to visualize data graphically in IDL is to plot to the display window.
- This is not the way you will create “publishable” graphics, but it is a good way to start looking at your data and it is a good way to learn how to display your data using the `plot` command.

A few things you should know

- IDL supports a system known as “Direct Graphics” which is built to a “device” - oriented model.
- What does this mean?? - you select a particular graphics device (screen, printer, PostScript) and then draw graphics on that device.
- This also means that you will have to switch the device to change how you visualize data.

Graphics

- The `set_plot` procedure selects a named graphics device.
- All subsequent graphics output is sent to the selected graphics device until another `set_plot` call is made or the IDL session ends
- `set_plot, device`
 - 'X', 'WIN', 'PS'
- Default is the screen display (`set_plot, 'x'`)
- The following would then display a plot to the screen: `IDL> plot, indgen(10)`

A few things you might notice ...

- IDL's default setting is to plot things with a black background using white font. This can be quite annoying at times and I will teach you how to change that in a bit.
- The window is labeled "IDL 0"
- We plotted 1 vector, but got an x vs. y plot.

Window Options

- Graphics window can be created directly using the window command or indirectly by issuing a graphics display command when no window is open. `IDL> window`
- The title bar of the window has a number in it (0 in this case) - this is the window's graphics window number index.
- You can open a new window with a graphics window index number of 1 by doing the following: `IDL> window, 1`
- You are allowed up to 128 different windows (but boy that would be confusing)
- IDL will assign window index numbers for windows 32-127 by using the `/free` keyword: `IDL> window, /free`
- If you don't explicitly open a new window, IDL will plot over whatever is on the current window.

Window options

- There are other options for setting, deleting, and clearing windows. Try the following:

- Open two windows (indexed 1 and 2) `IDL> window, 1`
`IDL> window, 2`
- Plot something in the current window (window 2) `IDL> plot, indgen(10)`
`IDL> wset, 1`
- Plot something in window 1 (using wset) `IDL> plot, sin(findgen(200))`
- delete window 2 `IDL> wdelete, 2`
- erase the contents of window 1 `IDL> erase, 1`

Name	Purpose
<code>window</code>	Create a new window
<code>wset</code>	Makes an existing window the current window
<code>wdelete</code>	Delete an existing window
<code>wshow</code>	Expose, hide or iconify an existing window
<code>erase</code>	Erase the contents of an existing window

Window Options

- You can also change the position and size of a graphics window.
- Use `xsize` and `ysize` to change the size of the window (these are in pixels)

```
IDL> window, 1, xsize=200, ysize = 300
```

- Use `xpos` and `ypos` to relocate the position of the graphics window. Windows are positioned on the display with respect to the upper left-hand corner of the display in pixel or device coordinates.

```
IDL> window, 2, xpos=75, ypos=150
```

Creating Plots

- The plot procedure draws graphs of vector arguments. It can be used to create line graphs, scatter plots, barplots, and even polar plots.

```
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] [, LINESSTYLE={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}]
```

- The plot command can be modified with over 50 different keywords!

Plotting a vector

- IDL will try to plot as nice a plot as it possibly can with as little information as it has.
- Try plotting: `IDL> plot, sin(findgen(100)*.2)`
- In this case the x or horizontal axis is labeled from 0-100, corresponding to the number of elements in the vector. The y or vertical axis is labeled with the data coordinates (this is the dependent data axis)

Plotting x vs. y

- Most of the time a line plot displays one data set (the independent data) plotted against a second data set (the dependent data).
- For example:

```
IDL> x = FINDGEN(21)*.1*!pi
IDL> y = sin(x)
IDL> plot, x, y
```
- x represents radian values extending from 0 - pi and y is the sine (x)

Customize Graphics Plots

- To label the x-axis, y-axis, and title use: `xtitle`, `ytitle`, and `title` keywords, respectively.

```
IDL> plot, x, y, xtitle = 'Radians', ytitle = 'Radians', $  
title = 'Sine Wave'
```

- By default, the plot title is 1.25 times larger than the x and y axes labels - there are a number of ways to change this:
 - To change the size of all the plot annotations use the `charsize` keyword.

```
IDL> plot, x, y, xtitle = 'Radians', ytitle = 'Radians', $  
title = 'Sine Wave', charsize = 1.5
```

- To change the character size of each individual axis use `[XYZ]charsize`.

```
IDL> plot, x, y, xtitle = 'Radians', ytitle = 'Radians', $  
title = 'Sine Wave', xcharsize = 2, ycharsize = 3
```

Modify line styles and thickness

- Use the `linestyle` keyword to plot the data with a different line style. For example to get a dashed line (instead of a solid line) use:

```
IDL> plot, x, y, linestyle = 2
```

- Use the `thick` keyword to change the thickness of the line plots. For example if you want to plot displayed with a dashed line that is three times thicker than normal try:

```
IDL> plot, x, y, linestyle = 2, thick = 3
```

Index	Line Style
0	Solid
1	Dotted
2	Dashed
3	Dash Dot
4	Dash Dot Dot
5	Long Dash

Symbols

- You can also plot your data using symbols instead of lines. Like the `linestyle` keyword, similar index numbers exist to allow you to choose different symbols.
- For example you can draw the plot with asterisks by setting the `psym` keyword to 2:

```
IDL> plot, x, y, psym = 2
```

- You can also connect your plot symbols with lines by using negative values for the `psym` keyword. To plot triangles connected by a solid line try:

```
IDL> plot, x, y, psym = -5
```

Index	Symbol
0	No symbol
1	Plus sign
2	Asterisk
3	Period
4	Diamond
5	Triangle
6	Square
7	X

Plot Style and Range

- You can also limit the amount of data you plot with keywords. To just plot data that lies between 1 and 3 on the x axis and -0.5 and 0.5 on the y axis try:

```
IDL> plot, x, y, xrange = [1,3], yrange = [-0.5,0.5]
```

- Sometimes IDL is lame and won't like your chosen axis range (because the chosen range is not "aesthetically" pleasing), use the [XYZ]style keyword to make IDL listen to you!

- You can change the way your plot looks by using the [XYZ]style keywords. For example to force an exact axis range use:

```
IDL> plot, x, y, xstyle = 1
```

Index	Axis Property
1	Force exact axis range
2	Extended axis range
4	Suppress entire axis
8	Suppress box style axis
16	Inhibit setting the y axis starting at value 0

Adding Lines to graphics

- Use plots to add lines to your plots.
- syntax: `plots, [x0,x1],[y0,y1],[z0,z1]`
- Example: add a line that spans the length of the x-axis and crosses through 0 on the y-axis.

```
plot, x, y, xrange = [0, 2*!pi],  
      xstyle = 1
```

```
plots, [0, 2*!pi],[0,0]
```

Tick marks, intervals, and names

- [XYZ]ticklen - controls the length of the axis tick marks (expressed as a fraction of the window size). Default is 0.02. ticklen =1.0 produces a grid, negative ticklen makes marks that extend outside the window.
- [XYZ]tickinterval - set to a scalar to indicate the interval between major tick marks
- [XYZ]tickname - a string of up to 30 elements that controls the annotation of each tick mark.

Plotting Multiple Data Sets

- Use the `oplot` command to plot multiple data sets on the same set of axes.

```
IDL> x = findgen(21)*.1*!pi
IDL> y = sin(x)
IDL> y2 = cos(x)
IDL> plot, x, y,
IDL> oplot, x, y2, linestyle = 2
```

- What if you have two data sets that require different axes?

Positioning

- You can also position the plot inside the window using the position keyword.
- position is a 4-element vector giving, in order, the coordinates $[(x_0, y_0), (x_1, y_1)]$ of the lower left and upper right corners of the data window. Coordinates are expressed in normalized units ranging from 0.0 to 1.0. Position keyword is never specified in data units.

```
IDL> x = findgen(21)*.1*!pi
IDL> y = sin(x)
IDL> plot, x, y, position = [0.2, 0.2, 0.8, 0.8]
```

Plotting with Multiple Axes

- Sometimes you have two or more data sets on the same line plots, but you want the data sets to use different y axes. It is easy to establish as many axes as you need with the axis command.
- The key to using the axis command is to use the /save keyword to save the proper plotting parameters.
- Try this: (*note the /ylog keyword is set to make the new axis have a log-scale

```
IDL> x = findgen(21)*.1*!pi
IDL> y = sin(x)
IDL> y2 = dindgen(21)^10
IDL> plot, x, y, position = [0.2,0.2,0.8,0.8], $
    xtitle = 'x', ytitle = 'y'
IDL> axis, yaxis=1, yrange = [.001,5E9],/save, $
    ytitle = 'other axis',/ylog
```

Multiple plots on a page

- It is much easier to use the `!p.multi` system variable to create multiple plots on a page. `!p.multi` is defined as a 5 element vector as follows:
- `!p.multi[0]` - number of graphics plots remaining to plot on the display. It is normally set to 0 meaning there is no more graphics plots remaining to be output on the display, the next graphic command will erase the display and start with the first of the multiple graphics plots
- `!p.multi[1]` - this element specifies the number of graphics columns on the page
- `!p.multi[2]` - this element specifies the number of graphics rows on the page
- `!p.multi[3]` - this element specifies the number of graphics plots stacked in the z direction ... but you must establish a 3-d coordinate system
- `!p.multi[4]` - This element specifies whether the graphics plots are going to be displayed by filling up the rows first = 0 or the columns first = 1

How to use !p.multi

```
x = findgen(21)*.1*!pi
y = sin(x)
y2 = cos(x)
; set !p.multi to create 4 plots, filling in the columns first
!p.multi = [0,2,2,0,1]
plot, x, y, linestyle = 0, title = 'Plot #1'
plot, x, y2, linestyle = 1, title = 'Plot #2'
plot, x, y, linestyle = 2, title = 'Plot #3'
plot, x, y2, linestyle = 3, title = 'Plot #4'
; set !p.multi to create 4 plots, filling in the rows first
!p.multi = [0,2,2,0,0]
plot, x, y, linestyle = 0, title = 'Plot #1'
plot, x, y2, linestyle = 1, title = 'Plot #2'
plot, x, y, linestyle = 2, title = 'Plot #3'
plot, x, y2, linestyle = 3, title = 'Plot #4'
```

Leaving room for Titles with Multiple Graphics plots

- When IDL calculates the position of the graphics plots, it uses the entire window to determine how big each plot should be.
- Sometimes you would like to have extra room on the display for titles and annotation.
- You can leave room with multiple plots by using the “outside margin” files of the !x, !y and !z system variables
- **Syntax:** `!p.multi = [0,2,2,0,1]`
`!y.omargin = [2,4]`

Adding text to Graphics

- The `xyouts` command is used to place a text string at a specific location on the display
- `xyouts` can be used to add a larger title to a line plot or to label data points on the graph.
- Syntax: `xyouts, 0.5, 0.9, /Normal, 'Graphics Plots', Alignment = 0.5, Charsize = 2.5.`
- You can specify the `x` and `y` location of the string with data coordinates or normalized (positional) coordinates.

Positioning Text

Embedded Command	Command Action
!A	Shift text above the division line
!B	Shift text below the division line
!C	Insert a carriage return and begin a new line of text
!D	Create a first-level subscript
!E	Create a first-level superscript
!!	Create an index-type subscript
!L	Create a second-level subscript
!N	Shift back to normal level after changing level
!R	Restore text position to last saved position
!S	Save the current text position
!U	Create an index-type superscript
!X	Return to entry font
!z	Display one or more characters by their unicode value
!!	Display the exclamation mark !

When plotting to the window

- More than likely your computer will have a 24-bit (or better??) graphics card. This means that when plotting to the window you want to be using a true color visual class (not pseudocolor or direct color). When plotting to the window we need to configure our device to use the 24-bit mode: `device, true = 24`
- You also need to set up how the contents of graphics windows will be refreshed when windows are moved in front of or behind each other (this is called “backing store”). To do this you need to set the retain keyword, I suggest using: `device, retain = 2`, which means that IDL is responsible for maintaining backing store.

Colors

- A color in IDL is composed of three specific values. We call these values a color triple and write the color as (red, green, blue or RGB). Where the red, green, and blue values represent the amount of red, green and blue light that should be assigned that color on the display. Each value ranges from 0 to 255, thus a color may be made up of 256 shades or gradients of red, 256 shades of green, and 256 shades of blue.
- This means that there are 16.7 million colors that can be represented on the display by IDL. (color triple that represents yellow is (255, 255, 0)).
- Two different color models can be used to specify colors on a 24-bit graphics display, and some of the direct graphics commands work differently depending on the model currently in use.

Color models

- The two different models are called decomposed color and indexed color. (decomposed on and decomposed off)
- When configuring the device decomposed = 0 (index model), decomposed = 1 (decomposed color model).
- By default, when you set the 24-bit Truecolor environment, it will be using the decomposed model, which means that you can set the color using an RGB triple directly or: `color = [255,255,0]` (yellow)
- I am going to teach you how to use the “index model” - because this is what we will need to use to make PostScripts.

Device Configuration

- To configure our device so that we can draw graphics to the screen and everything works out so that we don't get crazy flashing and unexpected colors do the following.

```
device, true=24, decompose=0, retain=2
```

- Here we are using the truecolor color class, using the indexed color mode, and allowing IDL to manage backing store.
- If you are like me ... you will put this into a procedure that you can run and you won't think about it again (until you want to make PostScripts)

Back to actually colors

- So how do colors work in IDL?
- Now that we have configured our device, we want to load in a color table.
- We'll use the IDL pre-defined color tables. If you're adventurous you can also create your own.
- The default color table in IDL is called B- W linear.
- We can look at the colors in this table using the `xpalette` or `cindex` procedures.
- To change the color table use: `loadct, # of CT :`
`loadct, 39`

Setting the background and plot colors in IDL

- IDL's default setting sets the background of the plot as black and the font as white. To change this use the background and color keywords. Set background and color to the desired color "index" in the selected color table.
- When using the B-W linear color table do the following:
- `plot, x, y, background = 255, color = 0`

Adding a Legend

- use the legend.pro procedure created by to add a legend to your diagram.
- See sounding.pro for basic syntax or legend.pro for all options

Producing Pretty Postscript Plots

```
SET_PLOT, 'PS'  
DEVICE, file='myfilename.eps', $  
XOFFSET = 0.8, YOFFSET = 1.0, $  
XSIZE=8, YSIZE=4, $  
/helvetica, /COLOR, /inches, /portrait, /ENCAPSUL
```

Execute plotting statements

```
DEVICE, /close  
SET_PLOT, 'X'
```