IDL Tutorial 1 -
Getting Comfortable with the Command Line

Getting Started:

To start IDL, type:
> sswidl
at the Linux prompt. This should start a program and lots of stuff should print to the screen, then you should see an idl prompt, like this:
IDL>

To complete the tutorial below, read the text below and enter the commands that are given by the IDL prompt. Sometimes, you will be asked to enter something at the LINUX prompt (">"), instead of the command prompt. To do this, open another terminal.

IDL can save all your interactive session by using the JOURNAL command. Start the recording process by typing the following command.
IDL> journal,'yourlastname.date.journal'
e.g., journal, 'winebarger.120531.journal'. When you exit IDL (or type 'journal' again with no filename), the recording session will end. After typing a few commands, check that the file has been created in the proper place and is recording your session.

There are different types of variables used in IDL (similar to fortran or C). The types you will use most frequently are integers (a 16 bit numbers with no decimals that can range from -32,768 to +32,767) and floating point numbers (a 32 bit numbers with decimal with 6 or 7 decimal precision that can range from $+-10^{38}$). Unlike fortran or C, you do not have to declare what variable something is, IDL just interprets the input. For instance, type:
IDL> a = 1
IDL> b = 1.0
IDL> help,a,b
This should print out something like:
A INT = 1
B FLOAT = 1.00000
which tells you that the variable A is an integer with a value of 1, while the variable B is a floating point number with a value of 1.0000. IDL decided that A was an integer and B was a float because you didn't include a decimal point when you declared A, but did when you declared B. (Note: In general, IDL variables are not case sensitive, though linux file names ARE case sensitive. I always use lower case for both filename and variables.)

Note the use of commas in the above "help" command. IDL is big on commas, get used to them!

To create a 1-D array, type:
IDL> x=[0,1,2,3,4,5]

IDL> help, x
What type of array does IDL think the variable x is? To print the contents of x, do:
IDL> print,x
This method of making arrays is ok if you just have a few numbers to include in the array, but what if you wanted to make an array that has 50 numbers? Then you can use the indgen command:
IDL> x = indgen(50)
IDL> help, x
IDL> print,x
The indgen command makes an array where every number in the array is its index, i.e., the 0^{th} number is 0, the 1^{st} number is 1, etc. Note that IDL starts counting all things from 0 instead of 1, so the last number in our array of 50 is actually 49.

If you wanted to make an array of floats instead of integers, then you could do:
IDL> x=[0.,1.,2.,3.,4.,5.,]
IDL> help,x
IDL> print,x
or
IDL> x= findgen(50)
IDL> help,x
IDL> print,x
indgen and findgen are helpful functions that we will use a lot this summer.

A way to make an empty array, with only 0.0 instead of numbers, you can
IDL> x = intarr(50)
IDL> y = fltarr(50)
IDL> help,x,y
IDL> print,x,y

[Let's make sure the journal command is working. Go to another window at at the linux prompt type
>ls
Do you see the file you created? Right now the file is empty, but after you end the session or type journal again, the file will contain all the things you typed at the prompt.]

There are many simple functions that are built into IDL and can operate on arrays. First, make a float array of 50 elements where every element is its index:
IDL> x= findgen(50)
Then, name a variable y that is just 2 * x.
IDL> y = 2*x
IDL> help,x,y
Because x was a floating array of 50 elements, y is a floating array of 50 elements too. You could print y to verify that it is indeed 2*x, but a better way to visualize the data is to plot them.
IDL> plot,x,y,title='First plot',xtitle='X axis',ytitle='y axis'
Verify that y is 2*x, i.e., that when the x = 10, that y = 20. In the plot command, x and y are arguments, i.e., you "pass" them into the command through the command line. IDL also uses keywords (like title, xtitle, and ytitle) to pass optional information. To see the

IDL documentation on the plot command, begin the IDL help by
IDL> ?
A new window should have opened. On the upper left corner, type "plot" into the search window. Then click on "PLOT procedure". In the right hand window, information about the plot procedure should have opened. This window contains information about the required syntax (note the things in the square brackets are optional), arguments, keywords, and an example. **Read through the plot procedure and type in the commands from the following examples: "Simple Plot", "X Versus Y Plots", "Axis Scaling", "Range Keyword", and "Plotting Symbols". Note: Ignore where it tells you to type "@plot01", if you go through these examples in order you should have no problems**.

Note that when you made the above plots, a graphical window was created and named "IDL 0" (the name of the window is on the grey box at the top of the window). You can open multiple plotting windows using IDL, and move between them easily.
IDL> window,0
IDL> plot,x,y
should reopen window 0 and regenerate the plot you have already made (but now without the titles). You can make another plot in a new window by
IDL> window,1 & plot,x,y*y
where the ampersand (&) is used to separate commands on the same line. In this plot, we are plotting y times y, verify that this is correct. To go back to the first window, we use the commend "wset",
IDL> wset, 0 & plot,x,y^2

The above plots were made to the screen, but you may want to save the plots as files to view later. One way to save a plot is to make an image of the plot, i.e.
IDL> write_gif,'myfirstplot.gif',tvrd()
will make an image in your current directory called "myfirstplot.gif" of the current plotting window using the tvrd (or "TV read") command. Note that the filename was given in singlequotations. To verify this plot was saved, click on the "Applications" in the tool bar on the upper portion of your screen, then click on "File Manager". This should show you all your files in your home directory. Scroll down to find the file called "myfirstplot.gif" and click on it. Does it look like the window?

Another type of file often used is a postscript file. To make a plot in a postscript file, you open a postscript "device" (file), issuing the plot commands, then close the device. See below. Note that I include some comments on the command line after the semi-colon, ";". IDL ignores everything after the semi-colon, you do not have to type it into IDL, they are there just to explain the command.
IDL> set_plot,'ps'
IDL> device, filename = 'yourname.plot1.ps'
IDL> plot,x,y,xtitle='xlabel',ytitle='ylabel',title='Your Title Here'
IDL> device,/close ;closes the file
IDL> set_plot,'x' ;sets all subsequent plots back to the windows.
Did it create a good file? Go back to the file manager and click on the postscript file. A program called "Gimp" should automatically open. Click import to see the file. Note that the plot is the same, but the colors are revered (black plot on a white background.)

After you view the file, close the Gimp windows and return to your IDL window.

Sometimes you may need to read and plot data from a text file. An example text file in your home directory and called "partydiv.txt". Look at this text file in your linux window to see what it includes:
> more partydiv.txt
As you can see, this text file contains information on the number of democrats and republicans serving in congress as a function of time.

To read this file, we will use the function "rd_tfile", i.e.,:
IDL> file='partydiv.txt'
IDL> r=rd_tfile(file,/nocomment,/auto)
The "/nocomment" tells rd_tfile to ignore any comments in the text file (any line that starts with a #) and "/auto" tells rd_tfile to break the rows up automatically.
Now the variable r contains the contents of the file.
IDL> help,r
IDL> print, r
Note that r is a string array so that all information stored in r are strings. Now we want to take the different rows, stored in r, and name them another variable, i.e.,
IDL> year = r[1,*])
IDL> help,year
If we just do this, year is a string array of [1,some number], but we really want year to be a float array of [some number], so do:
IDL> year = reform(float(r[1,*]))
IDL> help,year
Now, verify that the year has been saved as new variable in the format we want. Also extract the total number of house seat, number of democrat and number of republicans:
IDL> tot=reform(float(r[2,*]))
IDL>dem = reform(float(r[3,*]))
IDL> rep = reform(float(r[4,*]))

We would now like to make a plot with time (years) on the x-axis and the percentage of democrats or republicans on the y-axis. In this simple case, since we only have the years, not months or days, we could use the plot procedure:
IDL> plot,year, tot,xtitle='Year',ytitle='Total House Seats'
However we often have to include the months and days in other data sets (including the one in your task below). Therefore, lets talk about how to do it using "utplot", where ut stands for universal time.

First, open up your IDL help window and type utplot into the search window. The response should have been something line "No topics found". That is because utplot is a routine in the SolarSoft package (or ssw) not in the main IDL program. To get help on programs that are in the ssw package, use xdoc, i.e.
IDL> xdoc
In the search window type "utplot.pro". A new gui should open up and give you the information on utplot. Look through it quickly.
One thing to notice is that the date has to be in a string format. Above, we extracted the year from r and converted it to a float. Let's re-do that and keep it a string:

IDL> year = reform(r[1,*])
IDL> help, year
IDL> utplot,year,tot,xtitle='Year',ytitle='Total House Seats'
For the rest of this example, we will use plot, but you must use utplot (and outplot) in your task below.

Now let's plot the percentage of seats that were democrats and the percentage that were republicans. To plot the percentage that were democrats alone, we just need to
IDL> plot,year,dem/tot,xtitle'Year',ytitle='Percentage'
Now on top of this, let's plot the percentage of republicans:
IDL> oplot, year, rep/tot
This doesn't look very good. Two line plots on top of one another make it difficult to follow either.

One way to get around this is to use symbols instead of lines, i.e.
IDL> plot, year,dem/tot, xtitle'Year',ytitle='Percentage', psym = 1
IDL> oplot, year, rep/tot, psym =2
This isn't much better in this case (though might be useful in other cases) because you want to follow a line which is difficult to do with dots.

Another option is to use different linestyles:
IDL> plot, year,dem/tot, xtitle'Year',ytitle='Percentage'
IDL> oplot, year, rep/tot, linestyle =2
Note that "linestyle = 0" is a solid line and is the default so when we didn't indicate a linestyle with our original plot command, IDL assumed we wanted a solid line.

This is better, but we need to indicate which line is republicans and which is democrats. Let's do this using the legend command:
IDL> legend,['Democrats','Republicans'],linestyle=[0,1],/bottom,/right

A final way of making this plot would be to plot the two lines in a different color. We first run a program called "linecolors" which set the first 16 numbers in the color scale to different colors:
IDL> linecolors
Now we make our first plot. We want the data (the line we draw on the graph) to be a color, but we want the axis of the graph to be normal (i.e., white or black depending on the device). To do this, we first make a plot using the /nodata keyword. This sets up the axes, but leaves out the data
IDL> plot, year,dem/tot, xtitle'Year',ytitle='Percentage',/nodata
Now we can oplot the data in different colors
IDL> oplot, year, dem/tot, color=3
IDL> oplot, year, rep/tot, color=10
Experiment with different color numbers. On a piece of paper, note which color goes with which number (i.e., 3 = pink). To make a legend with the colored lines, we do
IDL> legend,['Democrats','Republicans'],linestyle=0,/bottom,/right,color=[3,10]

In the above example, we were plotting the percentage of republicans as a function of time and the percentage of democrats as a function of time. What would it look like if,

instead as a function of time, we plotted them as a function of one another:
IDL> plot,dem/tot,rep/tot,xtitle='Percentage Democrats',ytitle='Percentage Republicans'
Isn't that a big mess? This is one time where we want to use dots instead of lines:
IDL> plot,dem/tot,rep/tot,xtitle='Percentage Democrats',ytitle='Percentage Republicans', psym=1
Does this plot make sense to you? The time that the percentage of democrats is high, the percentage of republicans is low and vice versa. Why don't all the points lie on a single line? (Meaning, why isn't the percentage of republicans = 1-percentage of democrats?)

These two numbers are anti-correlated. We can show that mathematically by calculating the correlation coefficient. Do
IDL> print,correlate(dem/tot,rep/tot)
The answer should be close to -1. If the answer was +1 the two values would be absolutely correlated, if the answer was -1, the two arrays would be absolutely anti-correlated and if the correlation coefficient was 0, the two values would not be correlated at all.

This is the end of this IDL tutorial. Type
IDL> journal
to end your journal file. Go to the linux prompt to see the contents of the file:
> more yourlastname.date.journal

TASK:

Now that you have completed your tutorial, let's do some science!

Inside your home directory, you should find a file called "RecentIndices.txt". Use Linux to see what is inside the file and what every column of the data is. The first column is the year and the second is the month of the observation. The third column is the observed sunspot number, the 8^{th} is the observed radio flux, and the 10^{th} is the observed AP index (a geomagnetic index).

To complete this task, you must
1) generate an image (gif) files that contain the sunspot number as a function of time, the radio flux as a function of time, and the observed AP index as a function of time. Plot the lines in different colors and make a legend
NOTE: Use utplot to make these plots. To set the date for utplot, do
IDL> date=r[0,*]+'-'+r[1,*]

2) generate 2 postscript files, one that contains the scatter plot of the sunspot number on the x-axis and radio flux on the y axis and one that contains the sunspot number vs AP index. Calculate the correlation coefficient between the two values and include that information in the title of the plot.

Be sure to label all axes of your plots.
When you are done with this task, email the results (3 files) to
amy.winebarger@nasa.gov