

Introduction to IRAF

Based on “A Beginner’s Guide to Using IRAF” by J. V. Barnes¹

Overview

IRAF (Image Reduction and Analysis Facility) is a broad-based image-processing software package that was developed by the National Optical Astronomy Observatories for general use. It is available on line as a free download for use by individuals on PCs running Linux, as well as for use by observatories and universities.

If you have root access to a Linux machine, you are welcome to download and install IRAF and X11IRAF and use them to do the labs for this class. See <http://iraf.noao.edu>. If not, you should run IRAF in your own *astro1* account. You must connect to *astro1* by means of an X-window installation on a local computer, such as *Hummingbird Exceed* on the cluster computers or, for the strong of heart, *cygwin* (<http://x.cygwin.com/>) on your own Windows computer. For demonstration purposes, we will be running IRAF on the Department of Physics and Astronomy computer *astro1*, a Unix machine.

IRAF is essentially a library of programs, called tasks, that execute various functions. Associated with each task is a file of parameters whose values affect the execution of the task.

In the following descriptions, words that you might type will be printed in typewriter font, while names of programs, tasks, and files will be printed in *slanted* typeface.

Getting started

Before using IRAF on *astro1* for the first time, you must carry out the following steps while logged in to *astro1*.

1. Open a terminal window
2. Create a directory for IRAF (commonly called *iraf*; `mkdir iraf`)
3. Change to this directory (`cd iraf`).
4. Type `mkiraf` and answer the question about terminal type. The `xgterm` option is recommended if you expect to use the PCs in RO 305 or Unix workstations elsewhere. A file called *login.cl* will be created in the directory *iraf*. In order for *IRAF* to work properly, this file needs to be executed each time you start *IRAF*, which means that you need to start it while in that directory..
5. Open the file *login.cl* with a text editor. You will see many statements that begin with the comment character `#` and among them will be the statement, `set imtype = "hhh"`. It is strongly recommended that you remove the comment character so that this statement will be executed. If this statement is not present, insert it.

¹Available on line as <http://iraf.noao.edu/iraf/ftp/pub/beguide.ps.Z>

6. Type `xgterm &` to start a graphics terminal window. Or, to make a window with a scroll bar on the right, type: `xgterm -sbr &`. In that window, make sure *iraf* is your current working directory, and then type `c1` to start IRAF.
7. IRAF will prompt you with `c1>` .
8. When you are finished with IRAF, type `lo` and you will return to the *astro1* system prompt.

Basics

If you happen to use a different type of terminal from the one you designated with `mkiraf`, you can designate a new terminal type with the `stty` command. Type just `stty` to check the default terminal type, or type `stty` followed by a new terminal type. For example,

```
c1> stty 4010
```

To issue a command to the host system (Unix) from within IRAF, just type an exclamation point first. For example, to type page by page the file *rfits.hlp*, type `!more rfits.hlp` — except that this example is not very good because IRAF has the equivalent task, `page`.

IRAF remembers every command you type. If you need to recall what you have done or if you want to re-use a command without having to retype it (strongly recommended for complicated commands!), use the history editor. At the prompt, type `e` and then press the up arrow until the command you want to see or re-use is displayed. Backspace left or right within the command with the arrow keys, delete or insert characters as desired, and type `<Enter>` when ready to execute. If you are running IRAF on your own machine, you will have a modern version, which will give you access to previous commands with the uparrow key.

Packages and tasks

IRAF's tasks are grouped into packages. Except for the most basic ones, the packages must be loaded into the computer's memory before the tasks in them can be used. To load a package, type enough of its name so that IRAF can identify it uniquely. For example, to load the package `onedspec`, it is enough to type `one`.

The IRAF command prompt will now be the name of the most recently loaded package, such as `one`. Once a package has been loaded, it remains available in memory and does not have to be re-loaded. If you expect not to need a package any further, it is a good idea to unload it by typing `bye`.

Help

All the tasks and some of the packages have on-line manuals, which you can access by typing `help` (itself an IRAF task) followed by the name of the task. *Help* will allow you to page through the manual; you can quit at any time by typing `q`. If you want handier access to the help, redirect the output to a file with the `>` redirection character. For example, to create your own help file *rfits.hlp* on the task *splot*, type `help splot > splot.hlp`. Then, you can look at the file with an editor in order to go straight to the examples at the end, go backwards and forwards as many times as you like, create your own handy short help files, and so on. The help files take a little getting used to, but they are worth reading. For more information on *help*, type, `help help`.

Directories and files

For moving around among directories and handling files, IRAF has tasks that are in one-to-one correspondence with the basic Unix commands. For example:

- `mkdir nite1` Create a subdirectory called *nite1* in your current directory
- `cd nite1` Designate *nite1* as the default directory (or current working directory)
- `dir` List the files in the current default directory (`ls` also works).
- `dir nite1/` List the files in *nite1* if the default directory is one level above it. The usual wildcard characters, `*` and `?` can be used to select sets of file names.
- `path` or `pwd` Show the current default directory
- `back` Go back to (that is, make it the current default) the previous directory
- `cd ..` Go to the directory one level above the current default directory
- `del` followed by file name: delete a file
- `copy` followed by file or path names: copy a file
- `rename` followed by file or path names: rename or move a file

Parameters

Associated with each task and with some packages is a parameter file. For a package, the parameters in the file are used by all the tasks in the package. To list the parameters for a package or a task, type `lpar` followed by the name of the package or task, for example `lpar imstat`. As with other commands, you can redirect the output to a file if you want to record the parameter values you are using.

The parameters listed first must be specified each time the task is executed. If you do not specify a value, IRAF will prompt you for one. If it gives you a suggested value, you can either accept it by typing `<Enter>` or else type another one. Parameters listed in the parameter editor in parentheses, after the required ones, are not queried for when the task is executed; they are “hidden.” The current values are simply used.

To change or edit the parameter file for a given task, use the task *eparam*; type `epar` followed by the name of the task. You will see a full-window display of the parameter values. To enter or change a parameter, move the cursor with the arrow keys to the line of the parameter, type the desired value, and type `<Enter>`. As elsewhere in IRAF, the character `:` escapes to a command line:

To exit the parameter editor, type `:q`

To exit the editor without saving your changes, type `:q!`

To execute the task immediately, type `:g`

Executing tasks

In addition to the method mentioned above, you can execute a task by typing its name followed by any needed parameters. If you omit any required parameters, the task will prompt you for them. Hidden parameters can also be specified on the command line.

For example, the task `del` has the hidden parameter `verify`. If `verify = yes` the task will ask you for confirmation before a file is deleted. If you are sure you want to delete a file, and if `verify = yes`, you can cancel the verification by typing `del file.typ ver-` with the Boolean operator `-` indicating negation.

Aborting tasks

Any task can be aborted by typing `^c` (Ctrl-C, the Control and C keys held down simultaneously). In general, it is good practice to type `flpr` (flush process cache) after an abort.

Sometimes, `^c` does not work. Then, try `^z`. Strictly speaking, `^z` does not abort a task but only causes it to execute in the background. If you type `^z` while in IRAF, IRAF will go to the background and you will see the *astro1* system prompt. At this point, you have two choices.

- Type `fg` to return IRAF to the foreground; it will resume where it left off. If it has crashed, you have not solved your problem.
- Kill IRAF and start over. Steps:
 - Type `ps` to list the processes you are running. You will see a list of numbered items, of which one will be `c1.e`.
 - Note the number of that item, let's say `nnnnn`, and then type `kill -9 nnnnn` to kill IRAF²
 - Now type `c1`, reload your packages, and start over. Any work that you did not save to a file has been lost.

IRAF image files

The spectra we will be analyzing are stored in a format called FITS (Flexible Image Transport System). The FITS files we will be using have two parts: a text part called the header, and a binary part containing the data values associated with the pixels. IRAF tasks can open FITS files if they have the extension *.fits*.

The latest version of IRAF is equipped to handle FITS files directly, and these instructions will be written accordingly. However, you may occasionally create files in the old IRAF image file format, in which the header and the data portions of the image are stored in two separate files. In the STF (Space Telescope Format), which is recommended, the files have extension *.hhh* for the header and *.hhd* for the data or pixel file. In most tasks, you can refer to an image by the file name without the extension. If this procedure seems to cause trouble, however, add the *.hhh* or the *.fits* extension as appropriate.

²Rather than retyping the process number, double click it in the display and then press the middle mouse button. In Unix, highlighting texts automatically copies it to the system clipboard, and then clicking the middle mouse button copies it to the command line.

Displaying data

For the reduced spectroscopic data with which we will be working, the basic method for displaying data is the task `splot`. A recommended entry in the parameter file on the *options* line is `histogram`, which gives a step-function style of plot, a more realistic representation of sampled CCD data than a plot in which the data values are connected by straight lines. The task `splot` has a lengthy parameter file and many graphics commands. A short command list is available at the graphics command `?` and you may want to keep a help file handy also.

There are two ways to make a paper copy of a plot.

1. At the graphics cursor, type `:.snap eps1` to create an Encapsulated PostScript graphics file in landscape format. Then, you will find in your current directory a file with a name of the form `sgin n .eps` where n is a 4- or 5-digit number. You will want to rename this file with a more informative name, but the `.eps` extension is required.

To print the file from within IRAF, type `!lpr -P mh4lp myfile.eps` where `mh4lp` is the queue name of the large laser printer in MH 4014. You may substitute the queue name of any convenient printer.
2. At the graphics cursor, simply type `:.snap` or just `=` to make a dump of the graphics screen to a printer. In order for this command to work, you must previously have inserted in your `.cshrc` file the command, `setenv LPDEST ro3co` where, again, the name of your favorite printer queue may be substituted for `ro3co`. The `.cshrc` is the file that controls the behavior of your account on *astro1* and will be found in your home directory under Unix (not IRAF).

It's possible to save a plot to a binary graphics file, which can be recalled and edited later. This feature is helpful for complicated, labeled graphs that you may have prepared for display or inclusion in a paper. To save the current graphics display, type at the graphics screen: `:.write myfile` and to recall the file, type `:.read myfile`. A helpful task for displaying and reformatting saved graphics files is `gkimosaic`. Extensive lists of graphics commands common to all IRAF graphics tasks, such as the command `T` for putting text labels on graphs, are available with the commands (at the graphics screen) `:.help` and `:/help`.

Additional topics

Output redirection

If the output of a task would normally come to your terminal screen, you can direct it to a file instead with the `>` sign. For example, to save a long image header to a file called "file", type `imhead image.fits lo+ > file`.

You may also direct the output of a task to the input of another task with a pipe, denoted by the symbol `|`. For example, to display a long image header one screen at a time, type `imhead image.fits lo+ | page`.

File and image name templates

IRAF accepts the usual wild card characters, `*` and `?`, with their usual meanings. In addition, it recognizes lists of file names given as ranges of numbers or alphabetic characters in square

brackets. For example, `dir image000[1-9]*` will list files with names beginning “image0001” through “image0009,” while `dir [a-e]*` will list all files with names beginning with letters a through e.

All these techniques are means of creating image templates. In order to use a template, all the file names must already exist. Therefore, you cannot use templates for the output file names of copy and rename commands. For this purpose, you must use a substitution syntax as in this example, which substitutes the string `n2red` for the string `n1`:

```
rename n1*.fits %n1%n2red%*.fits
```

In order to operate on multiple files, it is sometimes helpful to create a file list, which can then appear, with an `@` character in front, as the input or output file parameter of any task. For example, suppose you have a list of files that you would like to rename. Create a list of their names in the file `files1`, and put the list of new names in the file `files2`. Then you can rename all the files with one command by typing, `rename @files1 @files2` as long as there is a one-to-one correspondence between the lists of input file names and output file names.

The task `files` generates a list of file names, one per line. Here are some of the more useful examples from the help for this task.

3. Generate a file list to be used to make a set of new files. The new file names will be the old file names with “_1” concatenated to the root, e.g., “root.x” would map to “root_1.x” and so on.

```
cl> files root.*//_1
```

...

5. Use string substitution to change the filename extension of a set of files to “.y”.

```
cl> files root.%*%y%
```

History and command logging

IRAF records all the commands you type. To print the last `n` commands, type `history n`. The task `history` has no parameter file, but it has a help file. I often log my work at the end of a session by typing `history 999` and redirecting the output of the `history` task to a file. It’s especially helpful to include in this file lists of task parameters generated by means of `lpar`. IRAF has a task `concatenate` for joining together two or more files.

To recall, edit, and re-use previous commands, type `e` and then use the up arrow key to move backward through the command history or, in IRAF 2.13 or 2.14, use the up-and-down-arrow keys.