

Introduction to IRAF

Brian Keeney

September 5, 2006

This handout will build upon what you learned in the “Introduction to Unix” handout to introduce you to the Image Reduction and Analysis Facility (IRAF), the software that we will be using to reduce data. In the descriptions below, I’ve planted several questions to make you think about what IRAF is doing for you. If you can’t answer a question, ask a neighbor and see if you can figure it out together. If you can’t, then please ask me to help you work through it.

Running IRAF

This section describes the steps necessary to start IRAF. You will always need to perform these steps for IRAF to run properly.

1. Open a terminal window.
2. Open an image viewer. There are two image viewers designed specifically for manipulating astronomical FITS images: `ds9` and `ximtool`. Both of these viewers are available on cosmos and IRAF can use either of these programs to interact with images that you take at the 24-inch. You can use whichever viewer you choose. I prefer `ds9` and will be able to answer any questions that are likely to arise on how to use it, but I have a passing familiarity with `ximtool` as well. You can open `ds9` by typing `ds9 &` in the terminal and `ximtool` by typing `ximtool &` (what does the `&` do?).

This step is not strictly necessary for IRAF to run properly; however, if you want to use IRAF to examine and manipulate images (which is what you will be doing 99% of the time) then you need to open an image viewer.

3. Change into the `iraf` directory that you created when we setup IRAF. It is a subdirectory of your home directory, so you should be able to get there by typing `cd iraf` from your home directory. Alternatively, you could get there from **any** directory by typing `cd ~/iraf`. If you run `pwd` from this directory, the result should look like `/raid/home/astr/ugrad/username/iraf`, where *username* is your username on cosmos (e.g., mine is `keeney`).
4. Run `ls` and verify that the directory `uparm/` and the file `login.cl` exist. They will be needed by IRAF at initialization.
5. Open a new terminal window by typing `xgterm &`. A new gray terminal window should pop up. We will always run IRAF from an `xgterm` because they are good at handling both text and graphics, and there are several IRAF tasks that you will be using which display graphics as output.

6. From the `xgterm`, type `c1`. This will start IRAF.

IRAF Basics

One thing that you should always remember is that IRAF and UNIX are not the same thing. IRAF is a program that runs from within UNIX, and although some of its tasks are similar to (or even have the same name as!) UNIX commands, they are different. To try to minimize confusion, I will call anything that you type into a UNIX terminal to make it perform a function a “command” (e.g., `cp`, `mkdir`, `cd`), whereas something that you type to make IRAF perform a function will be called a “task”. That said, IRAF has several built-in tasks that mimic the functionality of commonly-used UNIX commands. For example, the UNIX commands `pwd`, `ls`, `cd`, and `cp` will all work more or less as you would expect if you type them from within IRAF. However, the UNIX command for deleting a file, `rm`, will not work for deleting images from within IRAF; instead, you have to use the IRAF task `imdelete`.

Furthermore, the IRAF tasks that have the same names as UNIX commands do not work exactly the same way. The main difference that you will notice is that IRAF does not recognize many UNIX shortcuts. For example, while you can use the command `cd ~/iraf` to change to your `iraf` directory in a UNIX terminal, it will not work in IRAF (try it) because IRAF does not recognize the `~` shortcut. It does, however, recognize the `..` shortcut. Figuring out how IRAF tasks vary from UNIX commands is sometimes a frustrating experience, but the only way to do so is through a little bit of trial and error (and asking your TA if you are really stuck).

To make matters a bit more confusing, you can also invoke UNIX commands from within IRAF using a `!`. The `!` tells IRAF to use UNIX to execute the command instead of any built-in IRAF tasks. One situation in which this is particularly useful is if you have forgotten to open your image viewer before starting IRAF. If this happens, you do not have to open the image viewer from a separate terminal as you might expect; rather all you have to do is type `!ds9 &` (or `!ximtool &`) from the IRAF prompt (`c1>`) and the image viewer will open. Note that if you tried opening the image viewer from within IRAF and forgot the preceding `!`, the viewer would not open and IRAF would give you an error message.

Working with Images

The first thing that you have to do with your data from the 24-inch is change the extension of the images from `.FIT` to `.fits`. You can do this using the `cp` command. For example, if you are in a directory with a file named `sample.FIT`, you can rename it to `sample.fits` using `cp sample.FIT sample.fits`. Unfortunately, you have to rename images one at a time (i.e., `cp *.FIT *.fits` will **not** work), but since you only have a couple of images for the first lab, it shouldn't be too painful. For later labs I will provide a script to rename your images for you.

Since all of you will have slightly different image names for your data, I'm going to use *imasename* to refer to the name of whatever image you might be working with. In the case of the image in my `sbodata` directory (`~keeneey/sbodata`), *imasename* = `M16_Ha_3min`. Note that *imasename* = `M16_Ha_3min` and not `M16_Ha_3min.fits` because IRAF expects that the all images have a `.fits` extension (i.e., if you do not provide an extension in *imasename*, IRAF will assume that the image you want is *imasename.fits*). Note that if you do not change the extensions of your images from `.FIT` to `.fits`, IRAF will not be able to work with them.

1. Use `pwd` to determine the directory that you are in when IRAF first starts. It should be `/raid/home/astr/ugrad/username/iraf`.
2. If necessary, change into a directory where you have images stored.
3. Use the IRAF task `display` to display an image. The syntax for this task is `display imasename [n]`, where `n` is an optional argument telling the task what buffer to store the image in. IRAF is capable of storing up to 16 images at a time in different buffers. The square braces, `[]`, indicate that `n` is an optional argument; if `n` is not specified, then IRAF will prompt you for the buffer to store the image in. So, if you want to display an image, you can do so by typing either `display imasename` or `display imasename 1`, for example. Note that you can also display an image using the File menu in either `ds9` or `ximtool`.

IRAF Shortcut: You can shorten task names and IRAF will automatically use the task that you want if you have used a unique abbreviation. So, if you don't want to type out `display` every time, you could shorten it to `displ`, but not `d` because it is not a unique abbreviation. While this might not save you much typing in this particular instance, there are longer task names where this feature comes in very handy, much like the tab completion shortcut in UNIX (which IRAF does not support, by the way).

4. After your image is displayed, there are several tasks that you can run to examine its contents. The simplest is `imstatistics` (conveniently abbreviated as `imstat`). The syntax for using this task is `imstatistics imasename`. Use this task to examine an image. What information does it tell you?

Using `imexam` to Find Object Positions and Measure Seeing

A far more powerful task for analyzing images is `imexamine`. I'll normally refer to it by its abbreviated name of `imexam`. The syntax for using this task is `imexamine [imasename|n]`. Again, the square braces indicate that *imasename* and `n` are optional arguments; if you do not supply either of them then `imexam` will assume that you wish to examine the image loaded in the current buffer. Note that if there is no image loaded, using `imexam` without any arguments will cause the image viewer to crash. Alternately, you could specify an image name (*imasename*) or buffer (`n`) that you

wish to examine. If you specify a buffer, an image must already be stored in that buffer, but if you specify an image name and the image is not already loaded, `imexam` will display it for you.

1. Use this task to examine your image. After you hit `<Return>`, you should see that your cursor moves to the image viewer window and becomes an annulus. This is the easiest way to tell that you are in `imexam`. Click on the image viewer window to highlight it. Now `imexam` is waiting for input from you.
2. After highlighting the image viewer window, type `?`. This will cause `imexam` to display a list of all available inputs. The list will be displayed in the `xgterm` and should start with a line that reads `--IMEXAMINE COMMANDS--`. Highlight the `xgterm` and scroll through the help by pressing the spacebar. Find the input that will return the coordinates of the current cursor position in the image.
3. You can exit the help at any time by typing `q` while the `xgterm` is highlighted. After exiting the `imexam` help screen, highlight the image viewer window again. Notice that the cursor has turned back into an annulus.
4. Center your cursor on a star in your image and type the `imexam` input that will return the current cursor position. How accurately do you think you can determine the star's position with this method?
5. You can exit `imexam` at anytime by typing `q` while the image viewer window is highlighted.

You may have noticed that both `ds9` and `ximtool` already display your current cursor position, so finding the cursor position with `imexam` is a bit redundant. However, there are some functions that only `imexam` can perform, such as measuring the seeing in your images.

1. If you are not already in `imexam`, run it now.
2. Bring up the help screen and find the input that will create a radial plot.
3. Highlight the image viewer window and center your cursor on a star in your image. Now use the `imexam` input that will create a radial plot.

You will see a new window open that contains the radial plot for your star. In the center of the window you will find a plot of brightness (in counts) versus radius (in pixels) with a best-fit Gaussian plotted on top of the data points from the image. Basic information such as the version of IRAF that you are using, your username, the time and date that the plot was created, and the name of the image can be found above the plot. The star's centroid can also be found here; how does the accuracy of the star's position as found by radial profile fitting compare to your estimates from above?

All of the other pertinent information from the fit can be found in the row of numbers at the bottom of the window, which I will describe from left to right. The first number is the radius of the aperture used for fitting (why is it so large?). When a Gaussian is fit to the radial profile information, it also measures the brightness of the star; the next three numbers are the “magnitude” and flux of the star, followed by the mean sky counts per pixel in the aperture (what do I mean by flux in this context?). I put “magnitude” in quotes because this number has nothing to do with the real magnitude of the star; it’s simply calculated using the formula $mag = magzero - 2.5 \log flux$, where $magzero = 25$ by default (double-check that you can reproduce this “magnitude” from the flux displayed).

The fifth number is the star’s central brightness, I_c , as determined from the profile fitting (what are the units of I_c ?). This number could also be estimated from the plot in a pinch. When `imexam` is fitting radial profiles to the stellar image, it doesn’t assume anything *a priori* about the shape of the isophotes (contours of constant brightness). In other words, `imexam` does not assume that your star is circular, so it fits elliptical contours to the stellar image. The next two numbers are the ellipticity and position angle of these contours. The position angle values range from -90° to 90° , with 0° along the x-axis of the image.

The final three numbers are three different measures of the full-width at half maximum (FWHM) of the star, otherwise known as the seeing, or resolution, of your image. When determining the seeing in your image, you should measure the FWHM of several stars to make sure they agree. Stay away from stars that are saturated. The first FWHM measurement performed by `imexam` is calculated by fitting a Gaussian to the enclosed flux profile of the star (what does this mean?), the second measurement is calculated by fitting a Gaussian to the individual pixels (how is this different than the first method?), and the third measurement is a measurement of the FWHM of the star without assuming a profile shape. When determining the FWHM of a star, which of these three values do you think you should use?

I hope you are starting to realize what a useful and powerful tool `imexam` is. For quick reference later, the numbers on the bottom of the radial plot window are, from left to right:

1. Radius of fitting aperture
2. “Magnitude” of star
3. Flux of star
4. Mean sky counts per pixel in aperture
5. Central brightness
6. Contour ellipticity
7. Contour position angle
8. FWHM: Enclosed flux
9. FWHM: Individual pixels
10. FWHM: No assumed profile

Exiting IRAF

When you have finished, you can exit IRAF by typing `logout`. Now you should have all of the information that you need to reduce the data for the first lab. Good luck!