

## IRAF beginner's tutorial

[adapted from iraf's online tutorials]

This is a short exercise that should help acquaint you with the basics of IRAF. I will assume that you have created your IRAF directory, run "mkiraf" to create your login files, and that you have used "xiraf" to begin your IRAF session -- so you should have an xgterm and DS9 window open on your desktop.

Also, you should have copied the sample data files into one of your directories. Go into that directory now.

(N.B., in this tutorial the commands after the prompt, either "%" or "cl>" are meant to be typed by the user; the "#" sign indicates a comment.)

```
cl> cd ../sep16          # e.g., if you had put the images
                        # into ~/sep16/
cl> ls *fits            # check that im010 and im020 are there
```

The next step is to familiarize yourself a bit with some of the basic operations of IRAF. Tasks with similar/related functionality are grouped together in packages. Tasks, and sometimes packages, have parameter files that control their operation.

### 1) some basics

```
cl> package            # show what packages are currently loaded
cl> help images        # help for the package "images"
cl> help imheader     # help for task "imheader"
                        # (help will only be listed for tasks
                        # whose packages have been loaded)
    <space bar>      # go to next page of help
    q                # quit the help for this task
```

### 2) a look at task parameters, and viewing image headers with "imhead"

```
cl> lpar imheader     # list task parameters
                        # (note: you only need to type enough
                        # characters for the task to be
                        # identified -- try "lpar imh" and
                        # "lpar imhead")
cl> lpar imheader     # note query and hidden parameters - query
                        # parameters are listed first with hidden
                        # parameters following in parentheses

cl> imhead im010     # short header listing - what does it tell you?
cl> imhead im010 l+  # long listing
cl> ^ | page         # what did we just do? we recalled the last
                        # command and "piped" the output to the PAGE
                        # task, to look at it one page at a time
```

### 3) editing task parameters with "epar"

```
cl> lpar imhead
cl> epar imhead      # modify the task so it always types the long
                        # header listing - type yes after the parameter
                        # entry for "longheader" followed by a return
    :q              # exit eparam mode and "save" the new parameters
cl> lpar imhead     # was the edited parameter saved?

cl> unlearn imhead  # go back to the default parameter setting
```

4) viewing and editing image headers in batch mode with "hselect" and "hedit"

```
cl> imhead im010 1+ | page
                        # note the keyword "exptime"
cl> lpar hselect
cl> hselect im010.fits exptime yes
                        # see the help for hselect - it allows us
                        # to look at selected keywords from image
                        # headers
cl> lpar hedit
cl> hedit im*.fits notice "test data" add+
                        # add a keyword plus value to the headers of
                        # the two images - hit return after each query
cl> imhead im010 1+    # do you see the new keyword?
cl> hselect im*.fits notice yes
```

===

A few questions to review:

What have you learned about parameters? What is the difference between query and hidden parameters? What is the uparm directory? Do you understand the different syntaxes that we have been using in task executions?

===

A short exercise:

There are 2 images in your directory, im010 and im020, of the same field, but one frame has been slightly shifted from the other. We want to shift the second frame so it aligns with the first frame, and then average these two frames together.

```
cl> dir im*.fits        # note the two frames
cl> imhead im*.fits    # check to see if they are the same field
cl> unlearn display
cl> lpar display       # look at the parameters for the display task
cl> display im010 1    # load the first image into the image display
```

This is a good time to digress from IRAF a little and introduce DS9 ...

===

Using DS9, the image display tool:

First, display one of the images:

```
cl> display im010 1    # load the first image into the display
```

Notice the pull down menus at the top of the window on the left side. Pull each menu down and review the selections - use the left mouse button to pull down the menus (most of those selections are also available by clicking on the menus just above the image).

At the top left are rough coordinates of the current cursor position as well as an estimate of the pixel value at that cursor position. At the top right is a zoom box around the cursor position. To the left of the zoom window is a panner box, which shows you which part of the image you are currently looking at. Moving the turquoise rectangle inside that box will pan to a different part of the image.

The colourmap (usually greyscale) can be changed with the right mouse button. Drag up and down to change the contrast, left and right to change the brightness. Try dragging your mouse around to get a feel for how this works, noticing the different features in the image are shown at different contrast/brightness levels (e.g., object peaks

vs. detail at the level of the sky noise). You may wish to invert the colourmap; I find that easier to look at (change this from the "Color" menu).

DS9's default left-click behaviour is not very useful -- it puts a marker at the current cursor position. Sometimes this is good, but usually much better is to have the image pan to the current position upon a left-click. Do this by choosing "Pan" in the "Edit" menu.

Zoom is done through the menu. Also, try setting Edit > Preferences > Pan > "Pan then Zoom". Then if you click after panning to a new part of the image **\*\*without moving your mouse after the pan\*\*** you'll be zoomed in as well. It'll zoom in each time you click again, for several clicks, then zoom back to the original size. Very handy.

It is possible to load up to 4 "frames" (i.e., images, or image sections) on the display at once. Load the other image:

```
cl> display im020 2      #load the second image into frame 2
```

To switch between the two frames, click on the Frame menu just above the image (the lower menu, not the pull-down one). You can click "previous" and "next" to go between frames.

Since you have changed the contrast/bias, zoom, and pan of the first frame, the two probably won't match at all. Use Frame > Match Colorbars, and Frame > Match Frames > Physical, to match them up. Notice that the settings for the currently selected frame are kept and mapped onto the other frame.

By default, only one frame is shown at a time. But if you go to the Frame menu, you'll see options to "Tile" or "Blink" frames. You can view them side by side, or blink between them every second or so.

Lastly, you may want to print the DS9 output sometime. Just go to File > Print. You can either print directly to the printer, or (more usefully) save to a postscript file. Note that the resolution is pretty important; for good quality output you might want the dpi pretty high. Also, inverted colormaps tend to look better in printouts (and they save ink too). Finally, if you're saving to a file be warned that the default directory is probably not your current directory; click "browse" to choose the directory and filename you want.

DS9 has lots of features, which we don't have time to cover. You should play around and become familiar with it, since you'll be using it all year.

===

Back to the exercise...computing and correcting for the shift between the two images:

```
cl> display im010 1      # redisplay im010
cl> unlearn imexamine
cl> lpar imexamine      # we want to compute the shifts using this
                        # task interactively
cl> imexamine           # move the cursor into the DS9 window -
                        # notice that it has changed to a blinking
                        # circle - imexamine has put us into
                        # "interactive image cursor mode"
```

- a. put the cursor on some stars and press the "a" key - information will be printed in the xgterm window: the x-coord and y-coord values are on the first line with additional information on the second line - see the help page for imexamine for details

- b. with the cursor in the image window type "?" to see the cursor help  
- exit cursor help with "q" in the xgterm window
- c. what we would like to do is select 3 relatively bright stars  
and measure their positions in both frames and then compute  
the differences - we will use the average of the differences to  
shift the second frame so it aligns with the first  
  
mark 3 stars with the "a" key - space them over the image
- d. with the cursor in the image window, type "d" - you will be queried  
in the xgterm window for the next image - enter "im020" and frame "1"  
  
measure the same three stars in this image
- e. quit "imexamine" with "q" in the DS9 window

Compute the average shift, ie what shift do we want to apply to the  
second image so that it aligns with the first? The shifts that I computed  
are -0.53 and -1.68 - do you agree?

You can use IRAF as a calculator, by putting "=" at the beginning of  
a line. For instance:

```
cl> =5577.35-398.5
```

will give the answer "5178.85".

Now apply the calculated shift:

```
cl> lpar help
cl> help imshift sec=description # look at just one part of the
                                help page for the task that
                                we want to use

cl> help imshift sec=example
cl> unlearn imshift
cl> lpar imshift
cl> imshift im020 s020 ??? ??? # shift im020, and output to
                                a new file "s020". (Of
                                course, replace "???" with
                                the appropriate values.)
```

Now, blink im010 and s020 and see if we did a good job. Do you remember  
how to do this? Congratulate yourself if things look ok!

Let's try a little image arithmetic now. We have two frames, im010  
and s020, that we want to average. Let's do it two ways.

```
cl> unlearn imsum imarith
```

- a. cl> lpar imsum  
cl> imsum im010,s020 aver1 pixt=r calct=r option=average verbose+  
  
(or try "epar imsum", modify all of the parameters, and then  
type ":go")

[Note the concern here about the pixel type, both for the calculation  
and for the output image.]

- b. cl> lpar imarith  
cl> imarith im010 + s020 aver2 pixt=r calct=r v+  
(or try "epar imarith", edit all parameters, type ":go")  
cl> imarith aver2 / 2 aver2 # notice we are overwriting the

input image

c. [Hopefully the results are the same for both operations.]

```
cl> unlearn imstatistics
cl> lpar imstat
cl> imstat aver*.fits
```

Notice that when you change hidden parameters on the command line that they are NOT "learn"ed! How do you "learn" parameters?

===

Getting command history:

IRAF can redirect terminal output to a file, as well as pipe output from one task into the input of another task. There is also a history and recall mechanism.

```
cl> history # prints history tree
cl> ^ # recall and execute last command, can also
      # include number to execute any task in tree
cl> e lpar # recall last lpar command - allows you to edit
          # command line before executing with "return" -
          # use the arrow keys to move cursor, delete
          # or insert to the left of the cursor
cl> e # recall last command - use up/down arrows to
     # to go up/down history tree
cl> history 100 # look at last 100 commands
cl> history 100 > hfile # redirect output to a file
cl> page hfile # page the file
cl> history 100 | page # alternate method avoiding intermediate
                    # file
```

What is the difference between ">" and "|" ?

===

More on plotting:

Let's explore plotting in IRAF a bit. Some plot tasks are interactive and others are not. You can always replot whatever was in the last plot buffer and play with it.

```
cl> phelp plot # list help for "plot" package
cl> contour s020 # make a contour plot of your shifted
                # image - if it is not already the frame
                # that you are looking at in the ximtool
                # window, re-display it. Compare the plot
                # with the displayed image

cl> =gcur # recall last plot and look at it
          # interactively - we are now in "interactive
          # graphics cursor mode" - the cursor must
          # be in the plot window to accept commands

      :.help # these cursor options are available to
            # you in ALL plotting tasks in IRAF - they
            # are global cursor commands - quit the
            # help with a "q" in the text window

Z - place the cursor on a bright contour and type "Z"

A - put axis on zoomed plot

C - place the cursor on a feature and print out the current
    cursor positions
```



again

Now log out of the IRAF system.

```
cl> logout
```

If you started IRAF using the "xiraf" script (e.g., from the icon in your panel), then logging out should quit the xgterm and DS9 windows too. Otherwise, you can just quit them manually.

That's it!

===