ASTR 4100/6100

IRAF Tutorial

Before you do anything else, you must first make sure that IRAF is ready to use. Commands that you will type into a terminal screen are shown below in `courier` font. Whenever the directions tell you to type in a command, you should always hit the enter/return button after typing the command to make sure that it is issued and the computer knows to follow it. I will not tell you to hit "enter" every time.

0. Log in to the workstation with your phy3300-stNN account

1. Right-click on an empty part of the desktop and open a new terminal

2. In the terminal, type `ls`
This will list all the files and directories currently in the student account.

3. Make a directory called "iraf" by typing `mkdir iraf`

4. List the files and directories again by typing `ls`
This time, you should see the directory you just created in the list

5. Move into your new directory by typing `cd iraf`

6. To get IRAF ready for use, you need to do the following. This only has to be done once, and only if it has never been done before:
type `mkiraf`
when you see "Enter terminal type:" you should type `xgterm` and hit return
When the script has finished it will tell you that a new LOGIN.CL file has been created. You will now want to change a few things in that file.

7. Open the file that was just created by typing `vi login.cl`
You will see something like the following:

```
set    home        = "/home/phy3300-st01/iraf/"
set    imdir       = "/iraf/imdirs/phy3300-st01/"
set    cache       = "/iraf/cache/phy3300-st01/"
set    uparm       = "home$uparm/"
set    userid      = "phy3300-st01"

# Set the terminal type.  We assume the user has defined this correctly
# when issuing the MKIRAF and no longer key off the unix TERM to set a
# default.
if (!access (".hushiraf"))
   print "setting terminal type to xgterm..."
stty xgterm
```

Use the up/down arrows to move down through the file until you see the line:
#set stdimage = imt800

Use the up/down arrows to put the cursor on that line, and then press the "i" button on the keyboard to allow you to edit the line. Using only the arrow keys to move around, delete the "#" at the beginning of the line, and change *imt800* to *imt2048*
Next, look down a bit for the line that says:

#set imtype = "imh"

Again using only the arrow keys to move around, delete the "#" sign at the beginning of the line and change *imh* to *fits*

Those are the only changes necessary for IRAF to read our files properly, so press the "escape" button to exit out of editing mode and then type `:wq` (for *write* and *quit*)

Starting IRAF and opening an image display window

8. First open an xgterm (slightly different from a regular terminal) by typing `xgterm &`
(yes, the "&" is important, don't forget it)

9. IN THE XGTERM THAT YOU JUST OPENED, make sure that you are in the "iraf" directory. If you are not sure what directory you are in, type `pwd`
You should see in response: */home/phy3300-st01/iraf*
or a similar address based on your login name

10. Now start IRAF by typing (still in the xgterm): `cl`
You should see something that looks similar to the following:

NOAO/IRAF PC-IRAF Revision 2.15.1a EXPORT Mon Feb 21 18:54:16 MST 2011
This is the EXPORT version of IRAF V2.15.1a supporting PC systems.

Welcome to IRAF. To list the available commands, type ? or ??. To get detailed information about a command, type `help <command>'. To run a command or load a package, type its name. Type `bye' to exit a package, or `logout' to get out of the CL. Type `news' to find out what is new in the version of the system you are using.

Visit http://iraf.net if you have questions or to report problems.

**** Checking update status.... Your IRAF system is up to date*
**** Initializing SAMP .... No Hub Available*

The following commands or packages are currently defined:

apropos images. noao. proto. system.
dataio. language. obsolete. softools. tables.
dbms. lists. plot. stsdas. utilities.

vocl>

NOTE: If you see a message about there being no login.cl file, then either (1) you didn't follow steps 1-7 above, or (2) you are not in the iraf directory. Type `logout` , go back and do steps 1-7 or move into the correct directory, and then restart IRAF

11. Now you will want to open a display window so that you can look at the calibration files and images you will need. At the IRAF "vocl>" prompt, type `!ds9 &`
(Yes, you need to make sure that you have both the "!" at the beginning and "&" at the end, don't forget either one). You should see an SAOImage ds9 window open.

IRAF fiddly details and hints

--- IRAF will remember the commands you have recently issued. You can go through these commands (without retyping them) by using the command history. To scroll through your recently issued commands, use the up arrow. If you pass the one you want, use the down arrow to get back to it.

--- You can use the "tab" key to auto-complete a file name or directory address. Just type part of the name or directory and hit "tab" -- autocomplete will fill in as much as it can based on the uniqueness of the names and what you already typed.

--- All tasks in IRAF have associated parameter files. These parameters control the details for how a task will be carried out. In the steps directly below, you will use the task "display". To list the parameters controlling the task "display" and their current values, type `lpar display` To edit the parameters, type `epar display`

--- All tasks have help files that you can read by typing `phelp taskname` For the task "display", you would type `phelp display` . This is especially useful if you forget what a task does or what a parameter within a task controls. Note that the first line printed on the help page for any task gives the packages within which the task lives. Load the packages in order by typing their names (one at a time) at the IRAF prompt, and then you can run that task. *Packages only need to be loaded once per session*.

Displaying images - two different ways

12. Before you can work with any files in IRAF, you have to move to the directory where you have stored your files. In your xgterm window, with IRAF running, use `cd` to change to the correct directory.

13. Method number 1: use IRAF to display the image.
(Best for working with images directly in IRAF because many tasks will require you to interact with the displayed image.)
At the IRAF "cl>" prompt, type `display imagename.fit` and hit enter to accept the default image frame that IRAF suggests
(Shortcut: `display imagename` --- IRAF will automatically attach the ".fits" to the end
Shortcut 2: `display imagename 1` --- IRAF will display into frame 1 and won't prompt you for a frame number)

If you want to change the contrast or the stretch of the image, you will need to change those parameters within the IRAF task "display" and then re-display the image.

a.) Display the full intensity range of the image:

Type `epar display` to edit the parameters for the task "display", then use the arrows to move up and down among the parameters. When you get to the line for "zscale", type "no" to turn off the the compression of the intensity range, and make sure the line for "zrange" says "yes". To save the new parameters and quit the editor, type `:wq` and when you are back at the IRAF prompt, you can redisplay the image by typing `display imagename` (Shortcut: type `:go` to write and save your parameters and execute the task all at once from within the parameter editor)

b.) Display the full intensity range with a logarithmic stretch:
Type `epar display` and move among the parameters. Make sure "zscale" is still "no", "zrange" is "yes", and go all the way to the bottom to change "ztrans" from "linear" to "log". Save your parameters and re-display the image.

c.) Display a selected portion of the intensity range (e.g., 0 to 1000 counts):
Type `epar display` and move among the parameters. Make sure "zscale" is "no", "zrange" is "no", and at the bottom, set "z1" to your lowest intensity value (0, in this case) and "z2" to your highest intensity value (1000). Make sure that "ztrans" is what you prefer ("linear" or "log"), save the parameters and re-display the image. Try some different combinations of low and high intensity values and different scaling laws (linear vs log).

14. Method number 2: use ds9 to display the image without IRAF.
(Best for quick looks at images or working with images when you won't be using IRAF.)
In the ds9 window, go to "file" and choose "open" to search for and find your image file.

You can change the contrast and stretch by using the "scale" menu in ds9
a.) Display the full intensity range of the image: choose "minmax" from the "scale" menu
b.) Display the full intensity range with a logarithmic stretch: choose "log" and "minmax" from the "scale" menu
c.) Display a selected portion of the intensity range (e.g., 0 to 1000 counts): choose "log" or "linear" as appropriate, then choose "scale parameters" and type in your low and high values in the boxes. Hit "apply", then "close".

Investigating images

15. Determine the size of an image

```
imhead imagename
```

imagename [2048,2048][ushort]: the image has 2048x2048 pixels

16. Determine the minimum, maximum, mean, and standard deviation of the pixel values

```
imstat imagename
```

#	IMAGE	NPIX	MEAN	STDDEV	MIN	MAX
	imagename.fits	*4194304*	*1188.*	*28.98*	*1156.*	*23197.*

(Note: the numbers may not always line up properly with the column heads on your terminal, especially if your files have long names. However, they will always be in the correct order. The mean value will always be the third item listed, and the max value will always be the 6th item listed.)

17. Determine the minimum, maximum, mean, and standard deviation of the pixel values for a lot of images:

`imstat feb07*`
(*the * is a wild card that means here "any combination of characters after feb07"; examples: feb07.test.fit, feb07-001.fit, feb07-050.fit, but not vflat.fits)*

`imstat feb07-00?.fit`
(*the ? is a wild card that means here "any single character between feb07-00 and .fit"; examples: feb07-001.fit, feb07-008.fit, but not feb07-020.fit)*

`imstat @files`
(*the "@filelist" tells IRAF to look for a list of filenames inside the text file "filelist")*

18. Read all the associated info for your image

`imhead imagename lo+ | page`
(hitting the "return" button will take you down one line, the space bar will page down)

19. Make various plots from the image.

`imexam imagename`
(click somewhere on the ds9 image frame to activate the display window)
You will see a blinking circle. Use the mouse to move the circle so that it is hovering over an area of interest in the image (such as a star). Then type the various letters below (you do not need to click the mouse) to give shortcut calls to commands:

- `c` -- plot the intensity values in the current column
- `l` -- plot the intensity values in the current line
- `r` -- plot the intensity values radially from the current position
- `s` -- plot the intensity values as a surface map
- `e` -- plot the intensity values as a contour plot
- `h` -- plot the intensity values as a histogram
- `a` -- print measurements (position, flux, etc)

To quit at any time, simply press the "q" button. This will return you to the IRAF command prompt.

Each of the shortcut commands above has default parameters that control the command's behavior. To edit the parameters for the histogram plot above, for example, once you are at the command prompt, type `epar himexam` then use the arrows to move around and edit parameters, save and quit as usual. (Note: this is one case where the `:go` command will not work. You will have to use `:wq` to save and quit, and then reissue the `imexam` command).

20. Image arithmetic

```
imarith imagename1 + imagename2 addedimg
imarith imagename1 * imagename2 multimg
```

Exiting IRAF

To exit IRAF at any time, simply type `logout`

`****` **Now that you have gotten to this point, it's time to do the in-class worksheet!!**

Basic Linux cheat sheet:

Moving around in the file system	
Command	*Action*
pwd	"Print working directory" - show what dir you're in.
ls	List the contents of a dir.
ls -l	List the contents of a dir and show additional info of the files.
ls -a	List all files, including hidden files.
cd	Change directory.
cd ..	Go to the parent directory.
Examining files	
Command	*Action*
file	Determine the type of a file.
cat	Concatenate a file.
less	View text files and paginate them if needed.
Manipulating files and directories	
Command	*Action*
cp	Copy a file.
cp -i	Copy a file and ask before overwriting.
cp -r	Copy a directory with its contents.
mv	Move or rename a file.
mv -i	Move or rename a file and ask before overwriting.
rm	Remove a file.
rm -r	Remove a directory with its contents.
rm -i	Ask before removing a file. Good to use with the -r option.
mkdir	Make a directory.
rmdir	Remove an empty directory.

Basic data reduction steps - a skeleton tutorial for HLCO
(See also *A User's Guide to CCD Reductions with IRAF* on class website)

** Issue this command to get a copy of the "fixhead.cl" script: cp /home/bentz/fixhead.cl .
(the trailing "." is important, it will copy the script to the directory where you are located)

1. Fix the missing header information for your images.
Copy the script "fixhead.cl" into the directory with the data for the night you are working on.

Use your logsheet to note which images were taken for NGC5548 and for PG1323. Display each of those images to be sure that they are indeed numbered correctly in the spreadsheet (it's easy to make a mistake in the middle of the night, so you should always double-check).

Once you are convinced that the files match their descriptions in the logsheet, do the following:

epar hedit

The task "hedit" has the following parameters

```
images  =                    images to be edited
fields  =                     fields to be edited
value   =                     value expression
(add    =                yes) add rather than edit fields
(addonly=                yes) add only if field does not exist
(delete =                 no) delete rather than edit fields
(verify =                yes) verify each edit operation
(show   =                yes) print record of each edit operation
(update =                yes) enable updating of the image header
(mode   =                 ql)
```

In the "images" slot, you will give the filenames for each of the NGC5548 images.
In the "fields" slot, type "object" (quotes not necessary)
In the "value" slot, type "NGC5548" if the image is of NGC5548, or "PG1323" if it is an image of PG1323. Be sure to use all caps and no spaces.
Check that "add" through "update" have the same values as listed above. Type ":go" to run the program and accept the edits to the header information (hit enter when prompted with questions).

To check that the "object" tag has indeed changed in the header of an image, type
imhead filename lo+ | page *(the vertical bar is the 'pipe' symbol)*

Look for the header keyword "object" on the left and see that the value of the keyword has changed to whatever you told it to be.

When you have updated the "object" keyword for all the NGC5548 and PG1323-085 images, then you will want to run the "fixhead.cl" script by typing the following:

cl < fixhead.cl *(be sure to have a space on either side of "<")*

You should see lots of numbers scroll by, and when it finishes, you will know it has successfully completed if you see the following keywords added to the end of an image header (choose one of the NGC5548 images or PG1323 images to check):

EPOCH
RA
DEC
HJD
LJD
ST
AIRMASS
UTMIDDLE

Now you are ready to reduce your images.

2. Combine your bias images into a single master bias image for the night:

a. Edit the parameters for *zerocombine:*

```
input   =            @bias.lst  List of zero level images to combine
(output =            bias.fits) Output zero level name
(combine=               median) Type of combine operation
(reject =               minmax) Type of rejection
(ccdtype=                     ) CCD image type to combine
(process=                   no) Process images before combining?
(delete =                   no) Delete input images after combining?
(clobber=                   no) Clobber existing output image?
(scale  =                 none) Image scaling
(statsec=                     ) Image section for computing statistics
(nlow   =                    0) minmax: Number of low pixels to reject
(nhigh  =                    1) minmax: Number of high pixels to reject
(nkeep  =                       1) Minimum to keep (pos) or maximum to reject
(neg)
(mclip  =                  yes) Use median in sigma clipping algorithms?
(lsigma =                   3.) Lower sigma clipping factor
(hsigma =                   3.) Upper sigma clipping factor
(rdnoise=                  10.) ccdclip: CCD readout noise (electrons)
(gain   =                   5.) ccdclip: CCD gain (electrons/DN)
(snoise =                   0.) ccdclip: Sensitivity noise (fraction)
(pclip  =                 -0.5) pclip: Percentile clipping parameter
(blank  =                   0.) Value if there are no pixels
(mode   =                   ql)
```

Be sure to use the correct values for the readnoise and gain!!

b. Run *zerocombine*.

3. Subtract the bias structure from all your images using your master bias image.

a. Edit the parameters for *ccdproc*:

```
images  =                 *.fit   List of CCD images to correct
(output =            b_//*.fit) List of output CCD images
(ccdtype=                     ) CCD image type to correct
(max_cac=                    0) Maximum image caching memory (in Mbytes)
(noproc =                   no) List processing steps only?

(fixpix =                   no) Fix bad CCD lines and columns?
```

```
(oversca=                   no) Apply overscan strip correction?
(trim   =                   no) Trim the image?
(zerocor=                  yes) Apply zero level correction?
(darkcor=                   no) Apply dark count correction?
(flatcor=                   no) Apply flat field correction?
(illumco=                   no) Apply illumination correction?
(fringec=                   no) Apply fringe correction?
(readcor=              no) Convert zero level image to readout correction?
(scancor=                 no) Convert flat field image to scan correction?
(readaxi=                 line) Read out axis (column|line)
(fixfile=                     ) File describing the bad lines and columns
(biassec=                     ) Overscan strip image section
(trimsec=                     ) Trim data section
(zero   =            bias.fits) Zero level calibration image
(dark   =                     ) Dark count calibration image
(flat   =                     ) Flat field images
(illum  =                     ) Illumination correction images
(fringe =                     ) Fringe correction images
(minrepl=                   1.) Minimum flat field value
(scantyp=             shortscan) Scan type (shortscan|longscan)
(nscan  =                     1) Number of short scan lines

(interac=                   no) Fit overscan interactively?
(functio=             legendre) Fitting function
(order  =                    1) Number of polynomial terms or spline pieces
(sample =                    *) Sample points to fit
(naverag=                    1) Number of sample points to combine
(niterat=                    1) Number of rejection iterations
(low_rej=                   3.) Low sigma rejection facto
(high_re=                   3.) High sigma rejection factor
(grow   =                   0.) Rejection growing radius
(mode   =                   ql)
```

b. Run *ccdproc*. If you display an output bias frame after this step (e.g., b_feb07-001.fit), it should have basically zero counts for every pixel. Convince yourself that this is true. Use *imstat* to check this as well for your bias-subtracted biases.

4. Combine your dark images for each set of exposure times.

a. Edit the parameters for *darkcombine*. You can enter the filenames one of several ways:

i. type them individually on the "input" line, separated only by a comma

```
input   =    dark1,dark2,dark3 List of dark images to combine
```

ii. put the names in a list inside a file and have the task read the list file (the "@" is necessary for the task to realize that it is reading the names from a file)

```
input   =            @dark.lst List of dark images to combine
```

Choose an output name that makes sense (you'll probably have at least two sets of darks to combine, so make sure you know which is which):

```
(output =         dark180.fits) Output dark image root name
(combine=              average) Type of combine operation
(reject =               minmax) Type of rejection
(ccdtype=                     ) CCD image type to combine
```

```
(process=                   no) Process images before combining?
(delete =                   no) Delete input images after combining?
(clobber=                   no) Clobber existing output image?
(scale  =             exposure) Image scaling
(statsec=                    ) Image section for computing statistics
(nlow   =                    0) minmax: Number of low pixels to reject
(nhigh  =                    1) minmax: Number of high pixels to reject
(nkeep  =                       1) Minimum to keep (pos) or maximum to reject
(neg)
(mclip  =                  yes) Use median in sigma clipping algorithms?
(lsigma =                   3.) Lower sigma clipping factor
(hsigma =                   3.) Upper sigma clipping factor
(rdnoise=                  10.) ccdclip: CCD readout noise (electrons)
(gain   =                   5.) ccdclip: CCD gain (electrons/DN)
(snoise =                   0.) ccdclip: Sensitivity noise (fraction)
(pclip  =                 -0.5) pclip: Percentile clipping parameter
(blank  =                   0.) Value if there are no pixels
(mode   =                   ql)
```

Be sure to use the correct values for the readnoise and gain!!

b. Run *darkcombine* for each set of exposure times.

5. Subtract the dark current from each set of files with matching exposure times.

a. Edit *ccdproc* to now include the dark subtraction (leave everything else the same)

```
images  =          @img180.lst  List of CCD images to correct
(output =       d//@img180.lst) List of output CCD images
(ccdtype=                    ) CCD image type to correct
(max_cac=                    0) Maximum image caching memory (in Mbytes)
(noproc =                   no) List processing steps only?

(fixpix =                   no) Fix bad CCD lines and columns?
(oversca=                   no) Apply overscan strip correction?
(trim   =                   no) Trim the image?
(zerocor=                  yes) Apply zero level correction?
(darkcor=                  yes) Apply dark count correction?
(flatcor=                   no) Apply flat field correction?
(illumco=                   no) Apply illumination correction?
(fringec=                   no) Apply fringe correction?
(readcor=             no) Convert zero level image to readout correction?
(scancor=                no) Convert flat field image to scan correction?
(readaxi=                line) Read out axis (column|line)
(fixfile=                    ) File describing the bad lines and columns
(biassec=                    ) Overscan strip image section
(trimsec=                    ) Trim data section
(zero   =           bias.fits) Zero level calibration image
(dark   =        dark180.fits) Dark count calibration image
(flat   =                    ) Flat field images
(illum  =                    ) Illumination correction images
(fringe =                    ) Fringe correction images
(minrepl=                   1.) Minimum flat field value
(scantyp=           shortscan) Scan type (shortscan|longscan)
(nscan  =                    1) Number of short scan lines

(interac=                   no) Fit overscan interactively?
(functio=            legendre) Fitting function
(order  =                1) Number of polynomial terms or spline pieces
(sample =                   *) Sample points to fit
```

```
(naverag=                    1) Number of sample points to combine
(niterat=                    1) Number of rejection iterations
(low_rej=                   3.) Low sigma rejection facto
(high_re=                   3.) High sigma rejection factor
(grow   =                   0.) Rejection growing radius
(mode   =                   ql)
```

You can again use list inputs, wildcards, or typing individual file names into the input field. The example above has a list. Each of the files in "img180.lst" would have 180sec exposure times to match the 180sec dark frame, and the file names would be for the copies of these files that have already been bias subtracted (the "b_xxnn.fits" files). This list should include the bias-subtracted dark frames themselves and the bias-subtracted science frames.

b. Run *ccdproc*. If you display an output dark frame after this step (e.g., "db_feb07-060.fit"), it should have basically zero counts for every pixel (bias subtracted off and average dark current subtracted off). Convince yourself that this is true, and use *imstat* to investigate it as well.

5. Combine the flat fields for each set of filters. Use the bias-subtracted flats.

a. Edit the parameters for *flatcombine.*

```
input   =           @vflat.lst  List of flat field images to combine
(output =           vflat.fits) Output flat field root name
(combine=              average) Type of combine operation
(reject =            crreject) Type of rejection
(ccdtype=                    ) CCD image type to combine
(process=                  no) Process images before combining?
(subsets=                  no) Combine images by subset parameter?
(delete =                  no) Delete input images after combining?
(clobber=                  no) Clobber existing output image?
(scale  =                mode) Image scaling
(statsec=                    ) Image section for computing statistics
(nlow   =                   1) minmax: Number of low pixels to reject
(nhigh  =                   1) minmax: Number of high pixels to reject
(nkeep  =           1) Minimum to keep (pos) or maximum to reject (neg)
(mclip  =                 yes) Use median in sigma clipping algorithms?
(lsigma =                  3.) Lower sigma clipping factor
(hsigma =                  3.) Upper sigma clipping factor
(rdnoise=                 10.) ccdclip: CCD readout noise (electrons)
(gain   =                  5.) ccdclip: CCD gain (electrons/DN)
(snoise =                  0.) ccdclip: Sensitivity noise (fraction)
(pclip  =                -0.5) pclip: Percentile clipping parameter
(blank  =                  1.) Value if there are no pixels
(mode   =                  ql)
```

Be sure to use the correct values for the readnoise and gain!!

b. Run *flatcombine* for each set of filters.

6. Apply the flat field correction to all the images taken through that filter.

a. Edit *ccdproc* to now include the flat fielding -- also remove the dark file and turn off dark correction (IRAF will yell at you if you leave it on and feed it flat field images that weren't dark corrected already)

```
images  =            @vimg.lst  List of CCD images to correct
(output =         f//@vimg.lst) List of output CCD images
(ccdtype=                     ) CCD image type to correct
(max_cac=                   0) Maximum image caching memory (in Mbytes)
(noproc =                  no) List processing steps only?

(fixpix =                  no) Fix bad CCD lines and columns?
(oversca=                  no) Apply overscan strip correction?
(trim   =                  no) Trim the image?
(zerocor=                 yes) Apply zero level correction?
(darkcor=                  no) Apply dark count correction?
(flatcor=                 yes) Apply flat field correction?
(illumco=                  no) Apply illumination correction?
(fringec=                  no) Apply fringe correction?
(readcor=              no) Convert zero level image to readout correction?
(scancor=                 no) Convert flat field image to scan correction?
(readaxi=                line) Read out axis (column|line)
(fixfile=                     ) File describing the bad lines and columns
(biassec=                     ) Overscan strip image section
(trimsec=                     ) Trim data section
(zero   =            bias.fits) Zero level calibration image
(dark   =                     ) Dark count calibration image
(flat   =           vflat.fits) Flat field images
(illum  =                     ) Illumination correction images
(fringe =                     ) Fringe correction images
(minrepl=                   1.) Minimum flat field value
(scantyp=            shortscan) Scan type (shortscan|longscan)
(nscan  =                    1) Number of short scan lines

(interac=                  no) Fit overscan interactively?
(functio=             legendre) Fitting function
(order  =                 1) Number of polynomial terms or spline pieces
(sample =                    *) Sample points to fit
(naverag=                    1) Number of sample points to combine
(niterat=                    1) Number of rejection iterations
(low_rej=                   3.) Low sigma rejection facto
(high_re=                   3.) High sigma rejection factor
(grow   =                   0.) Rejection growing radius
(mode   =                   ql)
```

You can again use list inputs, wildcards, or typing individual file names into the input field. The example above has a list. Each of the files in "vimg.lst" would be a V-band image, including the individual V-band flat field images, and the file names would be the bias-subtracted and dark-subtracted versions (the "db_xxnn.fits" files; just bias-subtracted for the flats themselves).

b. Run *ccdproc*. If you display an output flat field image after this step (e.g., "fb_feb07-020.fits"), it should have values of basically 1 at every pixel (remember that you divide by a flat field, not subtract it). Convince yourself that this is true and use *imstat* to check it as well. You should see the dust donuts especially have gone away in the flat-corrected flat frames.

Congratulations! You now have reduced images! You should display them and check that everything looks ok before going on to make any measurements.