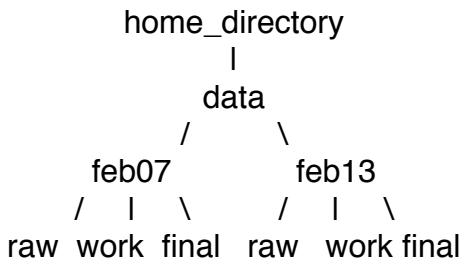
Basic data reduction steps - a skeleton tutorial for HLCO

(See also *A User's Guide to CCD Reductions with IRAF* on class website)

Before you begin, make sure that you have your data properly organized. This includes creating a backup copy of all your raw data that you can use to restore any files that you might accidentally edit or delete. I usually organize my data in the following way, and you can choose to use a similar scheme or something different that may make more sense for you personally:



All the reduction steps outlined below would be carried out on copies of the data in the "work" directory. The files in the "raw" directory would never be edited. The "final" directory is where I would put my final cleaned, reduced images and is the directory I would work in while making my measurements from the clean images.

Keep in mind that spaces are **BAD** in your file and directory names, because you will need to use them on the command line (and spaces on the command line usually separate commands). Use "." or "_" instead, or just mash all the characters together.

Refer back to the previous class handout for an introduction to basic IRAF commands, user interactions, and options.

Keep in mind: IRAF is not very good at overwriting images. If you do a step in this tutorial and make a new file, such as "bias.fits", then find that you need to change something and redo that step, make sure you delete the output file before trying to make it again. This will save you a lot of headaches. Just delete and then do over. The IRAF command "delete bias.fits" will do just that.

Each step of this tutorial will work with the results of the previous step. After subtracting the bias current, you will subtract the dark current *from the bias-subtracted files.* Think about your input files carefully (they are all described in each step) to be sure that you are reducing the counts at each step.

1. Combine your bias images into a single master bias image for the night so that you can remove any residual bias structure. The steps for this are below. You can choose to combine just the biases from your beginning of the night calibrations, but you could also look at the combination of **ALL** biases you took throughout the night (even any junk biases). Perhaps try both, but be sure to give each one a different output file name.

a. Edit the parameters for *zerocombine:*

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

```
input   =    bias1,bias2,bias3 List of bias 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   =           @bias.lst List of bias 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  =                mode) 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!! They can be found on the front page of the observing manual.

--> Remember that you may have to load the package holding *zerocombine* before you can actually edit and run it. Use phelp to tell you where the package lives.

b. Run *zerocombine*.

c. Examine the output combined bias image. Is the average pixel value what you expect? Does it look right? This is especially important if you combined all the biases from the whole night. There should be no "stars" (ghost images) in the frame, and if there are then you may need to change the "reject" type from "minmax" to something else (perhaps ccdclip), or increase the "nhigh" number for minmax rejection from "1" to something higher (perhaps 3 or 5). This will clip out any significantly high outlier pixels, such as ghost images.

2. Subtract the bias level and bias structure from <u>all</u> your images using your master bias image. Clay forgot to turn on the overscan for the camera, so we have to skip that part this year. The steps for the bias correction are below. If you made two or more biases in step one, then you need to choose the one you think is best for this step. Include the individual biases in the list of files to correct here.

a. Edit the parameters for *ccdproc*:

```
images  =          @im.lst      List of CCD images to correct
(output =      b_//@im.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=                   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=                  yes) 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*. This will take a while if you have a lot of images from the evening. When it has finished, display some of the output files to be sure that they all look correct. In particular:

- corrected bias images should have counts around zero (-20 <~ counts <~ 20)
- all corrected frames have typical pixel values that are lower by ~1000 counts now

(also use *imstat* to check the values for uncorrected and corrected files)

3. Combine your *bias-subtracted* dark images for each set of exposure times. The steps for this are below. You may want to combine a handful taken near the same time to compare with another handful taken at a different time during the night to see how stable the dark current was through the night. Or you could combine all the dark frames with the same exposure time.

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   =   b_dark1,b_dark2,b_dark3 List of dark images to combine
```

ii. choose an output name that makes sense (you may have several sets of darks to combine, so make sure you know which is which):

```
(output =         dark180.fits) Output dark image root 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  =                 mode) 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.

c. Look at the output files for each run. Again, there should be no ghost images if you were dithering properly. If there are, then you may need to change the "reject" type from "minmax" to something else (perhaps ccdclip), or increase the "nhigh" number for minmax rejection from "1" to something higher (perhaps 3 or 5). Always be sure that "nhigh" is smaller than the number of images you are combining.

4. **Subtract the dark current** from each set of corrected files with matching exposure times. The steps for this are below. If you found that the dark current changed noticeably throughout the night in the previous step, then you would subtract the dark frame from the images with the same exposure time taken around the same clock time, and you would do this step many times until all the science frames were corrected.

   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 with the same exposure time.

Note that *ccdproc* will not redo-the overscan and trim (it will check that this has been done with the stamp in the header and then do the new steps), so you can leave those lines as they were from the step before just as you see above.

b. Run *ccdproc*.

c. Examine some of the output dark frames after this step (e.g., "db_feb07-060.fit"), they 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. Examine **ALL** of the output science frames too to see how they appear. If there were any ghost images in the dark frame, you will see black spots (inverse stars) appear in your dark-subtracted image. If you see this, go back and redo your dark frames and be sure that you clip out any ghost image pixels.

5. Combine the bias-subtracted flat fields for each set of filters. The steps for this are below. Alternatively, if you have dark frames that match the exposure times of the flat fields, you want to correct them first. If you don't have dark frames that match, you could take one of your bias-subtracted combined dark frames and scale the pixel values by the exposure time ratio of the dark and your flats. Use this to subtract any small amount of dark current from your flats before combining the flat fields. Compare the results to the combined flats with no attempt at dark subtraction.

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=              median) 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.

c. Examine your combined flat field images and be sure they look correct.

6. **Apply the flat field correction** to all the images taken through that filter that we have corrected for dark current (the db_images). The steps for this are below. Do this for each filter separately.

a. Edit *ccdproc* to now include the flat fielding -- make sure to remove the dark file name from the parameters and turn off dark correction (this is because we did the final dark correction by hand instead of in ccdproc and it will get confused...).

```
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.

b. Run *ccdproc*. If you investigate an output flat field image after this step (e.g., "fdb_feb07-020.fits"), it should have the same mean value as before but a much lower standard deviation. 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. I recommend displaying an on-sky image in four different frames in ds9: the original image, the bias-subtracted image, the dark-subtracted image, and finally the flat-corrected image. You can view them side by side by going to the "frame" menu in ds9 and choosing "tile frames". The bias-subtracted image will look very similar to the raw frame, but the dark-subtracted and the flat-fielded images should look rather different (and better).

Appendix

Extra notes for the night of Feb07, where the x-axis was flipped after the calibrations were collected at the beginning of the night, but before the on-sky observations began.

To flip the x-axis of all the beginning of night files to match the rest of the observations, use the IRAF task *sflip*. Yes, it says it is for spectroscopy but it actually works for any CCD images.

First, create a list of all of the files that need to be flipped, one file name per line. Let's say that you named this list file "input_list".

a. Edit the parameters for *sflip*

```
        input = @input_list                Input spectra to flip
       output = flip.//@input_list         Output flipped spectra
(coord_flip = no)             Flip coordinate system?
 (data_flip = yes)            Flip data?
      (mode = "ql")
```

b. Run *sflip* and it will create new versions of the files with "flip." appended to the beginning of the file names so that you can easily find them. Display a flatfield image, before and after, side by side in DS9 so that you can compare the locations of the dust donuts before the axis is flipped and after it has been flipped. Then compare the flipped flatfield image to an on-sky image and again look for the locations of the dust donuts in the two images. If everything looks ok (dust donuts in flipped flatfield correspond to the locations of dust donuts in on-sky images) then you can use the flipped calibrations to proceed through the data reduction.