LRIS Multi-Slit Data Analysis

Eric Gawiser, 12/5/00

I. Preliminary reduction

(a) Philosophy for what follows: I want the statistics of the sigma spectra to be correct. Since we may use different instruments with different gains the safe thing is to turn ADUs back into electrons as our first step of processing. Note that *italics* describe IRAF routines and parameters and **boldface** describes UNIX commands and files. All non-standard iraf packages described here can be found in **~gawiser/iraf/bogus** or **~gawiser/iraf/wmkolris** and the **lamoffset150** code and examples of other relevant files are in **~gawiser/iraf/bogus/examples**. You will need to adapt the relevant portions of **~gawiser/iraf/loginuser.cl** in order for IRAF to see those packages. Most of the specific numbers given here will only work for the 150 line/mm grating on LRIS - if using a different grating/instrument you'll need to adapt the numbers but the methods should be viable.

(b) We will OV-subtract, gain-correct, and trim all images (2-D multislit spectra, arc-lamps, flats, and zeros/darks) using *nglccdproc* (ngl stands for no-gain, lris). Always work in a directory other than the raw directory since IRAF will overwrite files in defiance of all protections. *nglccdproc* calls *nglrisbias* which uses the ratio of input parameters *gain_left* and *gain_right* to set both to *gain_left*. *nglccdproc* then multiplies both by *gain_left* and sets header card *gain*=1.0. You can determine gain_left/gain_right empirically using a domeflat or skyflat but not from halogen flats of course. Current LRIS gains are listed as 1.97 (left), 2.10 (right) on their web page.

(c) First *nglccdproc* the zeros/darks. Hopefully, it will complain that **instr_lris_img.dat** is not found - ignore that unless it doesn't occur, in which case you should remove that file from the current directory since all it does here is cause trouble. If they appear relevant, zeros/darks can be subtracted from the rest of the frames using the *zerocor* and *zero* parameters of *nglccdproc*, or by first running *nglccdproc* without zero correction and then saying *imarith frame - zero_bias frame* if there is a uniform bias *zero_bias*. It is also possible to subtract a "dark" during *bogus* but that dark will not be subtracted from the flat at that point. Even a couple-pixel offset is worth fixing - especially if it's not uniform. If it is, it will be subtracted when *bogus* calls *background* and this uniform offset should not significantly affect the uncertainty of sky subtraction. Even a uniform offset would affect the flat but not the normalized flat produced by *response*. Think about how accurate you need image/flat to be... Either way, go ahead and run *nglccdproc* on all other object frames now.

(d) Although updating of headers is now handled automatically, if you need to edit single header cards, the easiest way is with *hedit header_card value=new_value add+ update+*. Don't ask me why but you need to say *add+* in order to add a new header card **or** update an old one. An alternative that's good for altering a lot of the header at once is the unix command **hdedit -novi**.

(e) All subsequent IRAF routines must now be fed *gain=1.0*. You'll also need to adjust *saturation* in routines that have it to reflect saturation in electrons instead of ADU. The readnoise is altered by combining images; the program that we will use updates that in the image header, but you still need to remember to check the header of the combined image and enter that *rdnoise* value (different for each field if they have different N_exposures) in the corresponding iraf parameter for routines like *apall* that use it. Always check IRAF routines for gain, rdnoise, and saturation because many routines have these parameters although they may not always use them. This really shouldn't matter until doing the statistics of a sigma spectrum in *apall*, though.

II. Combine and flat to get slitlets

(a) There are 3 input files that *bogus* requires:

i. **LIST.OBJ** Syntax is one imagename per line; use a carriage return so that the file ends on a blank line. **vi** and **emacs** differ here, but if you **more** the file and see a blank line, *bogus* may not work. If *bogus* complains, the presence of extra blank lines or lack of any is the likely problem. This applies to the other two files. The lack of a blank line can make it do the last file twice so watch for that during verbose output.

ii. **LIST.SHIFT** Syntax is: **x_shift y_shift** where the first line should be **0 0** and every other line is the shift needed to line up with the first one. Lines here directly correspond to the filenames on the same line of **LIST.OBJ**. If you dithered up (i.e. objects moved down in the slits), then put an e.g. +5 in y_shift here because you will need to move the current image up 5 pixels to put the spectra on top of each other. If you get the sign wrong you'll have multiple spectra in the output which should be pretty obvious. To determine the shifts, use **ds9** or **ximtool** to look at the multislit images, pick a few skylines and a few bright spectra to compare the coordinates of. Try to make them roughly symmetrical about the center but spanning across the whole image to see if your results are consistent. Use *implot, :c column_num, :l line_num, w e e* to zoom, and *space bar* to get the precise centers of these spectra and lines.

iii. **LIST.SLITLET** Syntax is: **objectname y_min y_max x_6300** Put the bottom-most slitlet on the first line and move up from there - then the output mosaics of *bogus* will look right. You'll want to use *implot* on the first (i.e. shifted 0 0) image to determine the x value of the 6300Å skyline and y_min and y_max. I don't think any harm is caused by choosing a different skyline such as 5577Å but whatever you choose will be shifted to column 500. You can tell it what you choose by *epar bogus*, *lambda*=6300. Look at a copy of the slitmask while you're doing this and try to keep as much sky as possible if the object is near one end of the slit or if it's a really short slit, otherwise be conservative so we don't get edge effects. I sometimes have to run *bogus* over again after revising **LIST.SLITLET** to eliminate a bad edge creeping in, but for short slits you have to push it. General rule is about 2 pixels into the plateau of the slitlet is far enough from the edge. The lack of a blank line at the end of **LIST.SLITLET** can make *bogus* skip the last slitlet. Oops!

(b) We also need to make a flat and choose an arc_lamp. For the arc_lamp, just choose the best-looking one that was taken at the same telescope position as the data. For the flat, check for cosmic rays. If none are found, just add the flats. Note that if there are $\geq$ 2 flats you can *imcombine* them using *combine=average,reject=crreject* (or *combine=median* for $\geq$ 3).

(c) Get ready to run *bogus*.

i. *bogus* calls *response*, which "normalizes" the flat so that it only represents pixel-to-pixel variations. You can have it fit the response interactively, which is recommended! (*func=spline3 order=15* is a reasonable default; *spline3* works better than *legendre* or *chebyshev* for this.)

ii. Proper statistics requires multiplying the averaged images output by *imcombine* by the number of input images and updating the ELAPTIME header card to account for that. I have modified the factory version of *bogus* to do precisely that, and to update the RDNOISE header card (roughly equal to 6.5 for a single LRIS image) to equal RDNOISE*sqrt(N_exposures), where N_exposures is found in the NCOMBINE header keyword added by *imcombine*. You will want to *epar hedit* and set *verify=no,show=yes,enable=yes* in order to allow header updating and have a record printed to screen but to avoid having to approve each action.

iii. During *imcombine*, we scale by ELAPTIME so that sigma clipping makes sense - the images should be similar in counts/s not total counts if they have different exposure times. One danger is varying sky level - we use *reject=sigclip* to get an empirical sense of how much this is varying; as long as we don't reject values less than 2 sigma from the mean (*epar imcombine* and set *lsigma=hsigma=2*) this should allow for reasonable sky variation but still nail residual cosmic rays, 0 values, or similar calamities. We purposely do no rejection on the .cr frame to keep a record of where cosmic rays and other nasty features were.

iv. *epar bogus* to make sure you're ready. Enter flat and arc_lamp filenames. For a quick runthrough turn off interactive response fitting and interactive background subtraction. This works OK for bright objects in long slitlets without bright interlopers, but if any of those criteria are wrong you will want to do background subtraction interactively. Note that some options of *bogus* such as "quick reduction" don't really work. Setting *displ=yes* makes the mosaic come on screen when *bogus* is finished.

v. *epar background* - make sure it is using *axis=2*. I recommend *niterate=1*. Rerunning background later with *axis=1* is a cute way to look for emission lines, according to Steve Dawson.

vi. Make sure *xzap* has *min,max* = -+ 127100 (almost saturation). This is a Mark Dickinson custom routine that you're officially supposed to get his permission in order to use.

vii. Important note: *bogus* comes from the factory with an inherent flaw, that of using asymmetric values of *low_reject* and *high_reject* during *background*. I recommend 2 sigma for each and have modified the code to do this. The rationale for asymmetric values is that there are positive sources out there that you want to reject but no negative sources. But once you get rid of any bright sources, the sky outline is noise dominated, and using asymmetric rejection criteria creates a bias, in this case an underestimation of the true sky level. You know that things went well if the residual skylines in the mosaic stay equally bright when you invert the color table - this means that they are just regions of higher noise but have been subtracted well leaving a zero mean. My modification produces better results than the factory version.

(d) Run *bogus*.

i. If you're fitting the response interactively (which is recommended), the idea is to match how well the response was fit for the spectrophotometric standard in making a sensitivity function (see below). If the fit is indistinguishable from the data on the full window plot, it's probably sufficient, although you often have to zoom around the peak to decide what order is appropriate. Be careful with *func=spline3* not to go too high in order since it can fit almost anything, but order 15 to 20 is reasonable here!

ii. It's best to do the background subtraction interactively because our objects are too dim to get deleted by *background* and therefore they affect the estimation of the sky level near them! This is an obvious statistical bias that reduces their flux, which we really don't want. So the trick is to know what y value you expect the object to be at (for a quasar you can see it but $m_i = 24$ galaxies are usually not visible before sky subtraction). This y value is calculable from the y_min,y_max,y_object output by e.g. **autoslit** - those y's do not correspond but the fractional location will be right. The expected y value will shift due to dithering so get that right for each frame that you're handling background on. Use *fitcolumn = (x_6300 + 19) (x_6300 +29)* to look at ~10 columns not including sky lines but beyond possible Lyman alpha forest absorption. For dim objects, you need to sum over more columns for the objects to show up versus sky fluctuations, so try *fitcolumn = (x_6300 + 19) (x_6300 +100)* and *fitcolumn = (x_6300 + 19) (x_6300 +190)* and see what looks best. Use *s s* to pick fit sections. Make sure that 5-10 pixels on each side of the expected y value are **not** part of the sample used for sky estimation, and delete nasty slitlet ends and bright interlopers, and you will significantly improve the results versus running *background* in its automated form. Deciding what order polynomial (I recommend avoiding splines here) to use in sky subtraction is subtle. The built-in values for *bogus* as a function of slitlet length are a good guide but are often a bit high. For long (20″) slits don't be afraid to use order 6 or higher, but for short (< 10″) slits order 3 is probably the max you can justify. Edge effects including alignment star bleeding could justify a higher order but it's better to just section those out. You can use the rms as a guide but make sure that you believe the interpolation across the object location - that's all we really care about, after all! A clue that the order is too high is oscillation of the 2nd derivative - we don't really expect the illumination pattern to have any inflection points within a slitlet. When you're satisfied, hit *q* then *return* to move on. The second version of each (**fzt**) is the CR-unzapped version. I just hit *return* and let background subtraction be automated on those, sort of as a control.

III. Extract the spectrum (do this section once for each object)

(a) Compare different versions: *dispbog objectname* automatically pulls the relevant reduced slitlets into an **ximtool** (or **ds9**) for you. objectname.fringe_s has no fringe subtraction, and .v1_s and .v2_s are different fringe subtraction algorithms. The fourth entry, .cr_s, has not been cosmic ray subtracted. The _s means shifted so that the x coordinate is the same function of wavelength for all slitlets, as displayed in the mosaic. The mosaic is therefore a very quick place to look for emission lines at a given redshift and notice any serendipitous clusters of bright objects all at a common redshift. There are also these neat .v1s_s and .v2s_s frames that are equal to the .fringe frame on the blue side but switch into the .v1_s or .v2_s frames on the red side where you'd really like to get rid of the fringes.

(b) Pick the version you like best - for $z \leq 4$ fringing is not a critical problem, although it affects the redder metal lines you're hoping to see. *dispbog* has been altered versus the factory version to display the .v1s_s version which is the best fringe/no-fringe composite and is usually the best one to use. The fringe subtraction algorithms tend to also subtract a portion of the object's flux, hence the .fringe_s version is the best to look at to determine if the object's spectrum keeps rising (past x=900 for 150 line/mm grating) i.e. it's at low redshift. Dan Stern recommends .v1_s over .v2_s in general if you want to extract a spectrum that will be better at the red end. Type *imcopy objectname.fringe_s objectname* to place your favorite version as the slitlet that we will extract the spectrum from.

(c) Using *apall* is a bit of a black art.

i. epar apall to check parameters - recommendations: *format=multispec, interac=yes, b_grow=1, t_nsum=200, t_step=50, t_order=1, backgro=fit.* All other early yes/no parameters are yes except for *resize=no.* The t_ stuff is for tracing. Don't forget to update *rdnoise* to match the value in the image header.

ii. Dan Stern recommends using *lower=-3.5, upper=3.5* for an aperture for LRIS and says that the "optimal" extraction usually doesn't make much difference. I agree - the unweighted version of the spectrum is usually quite similar to the variance-weighted extraction because they're sky-noise dominated, but the weighted version should be slightly better. The aperture should be bigger for extended objects or bad seeing - in general we want the fwhm of the aperture to equal the fwhm of the seeing.

iii. Another parameter to check is *skybox=5.* This performs boxcar smoothing of the sky background before subtracting it. Since we already ran *background* the mean sky level is gone but we can still accomplish something by smoothing the noise before subtracting it. The drawback is that skylines are broadened a bit, hence a higher value of *skybox* is not recommended. Also, since our sky noise is dominated by the sqrt of the sky counts in each column of the aperture originally, improving the background level estimation this way is only a minor improvement in sky subtraction. Nonetheless it is a robust improvement so I recommend it - setting *skybox=5* is enough to smooth out rough edges in the extracted spectrum before smoothing it - this smooths the sky noise but leaves the resulting object spectrum at full resolution.

iv. OK, try *apall objectname.* Typing *:line 560* takes you to column 560 and then *:nsum 80* makes it sum over 80 columns to see if any objects are there. I recommend trying those numbers and also *:line 600, :nsum 180* for z=4 work. If you dithered, remember that this is now a combined image, so the object y position should equal the minimum object y position from the individual dithered slitlets.

v. *b* checks background, *w a* shows whole background region, *t* erases the sample regions, *s s* chooses a new sample region and can be used repeatedly, *f* refits, and *q* declares victory and moves on. Remember, we have already run *background* once as part of *bogus* so the sky should have been subtracted. This is a chance for us to fix any clear under- or oversubtractions but only a low order polynomial (1-2, at most 3) is justified since we're probably looking at mostly noise.

vi. Hit *q* to trace. You definitely want to trace the aperture interactively. Notice that dim objects are lost in the Lyman alpha forest (and definitely past the Lyman break) and at the red fringy end. Look at the slitlet image again to try to figure out if anything beyond a horizontal line is appropriate - at most there's usually a couple of pixels different from the left to the right end so order 2 will suffice. You may need to use *s s* to section and to delete several points using *d* in order to make the trace go where you want. For dim objects you should pretty much override what *apall* tries to do and force it to trace a horizontal line at the y_value of the aperture you chose at the beginning or at the apparent y_value of the object in the slitlet image.

vii. For faint objects, try several different extractions and see which spectrum looks best in terms of signal/noise. You might as well go through this process once on each serendipitous spectrum you find as well in case there's something interesting. Beware *iraf*'s tendency to overwrite things when you don't want to and quit without overwriting when you did want to - some renaming of spectra will make this go more smoothly.

(d) When the spectrum is extracted, display it using *splot objectname.ms*. You can make *apall* review the extracted spectrum but it's not interactive. In *splot* you can blow up regions using *w e e*, use *o g* to overplot, and use *s 3* to smooth by e.g. 3 pixels. More than 5 pixel smoothing degrades the resolution significantly (any smoothing degrades the resolution somewhat) but can be useful for looking for a Lyman alpha decrement.

IV. Wavelength and flux calibrate

(a) Making a wavelength solution (1-3 times per slitmask)

i. *apall objectname.lamp_s refer=objectname out=lampref_objectname int- fit- tra- rec- back-* extracts an arc spectrum from the arc_lamp slitlet in exactly the same way that you just extracted the object spectrum. Now *epar identify* and put in the list of arc lines you want to use e.g. arc_lins.dat. For 150 line/mm grating, make sure that 2x5460 is in there - this is the only prominent second order line. Run *identify lampref_objectname*, use *w e e* to zoom, *m* to mark a few line IDs by comparing with the arcline plots from the LRIS website and entering your guess at their wavelength, *l* to go to the list of arc lines and have IRAF guess at the rest, *f* to fit, and *h* to give you a plot of residuals that should be a straight line. Be careful - Xe 8819 is **bright** but not on the arc line plots and is therefore easily confused with Ar 9122. Use *z* to check all of the lines fit by IRAF, hit *d* to delete a blend or really weak line and *n* to move on to the next one. Once the rms is less than a pixel (5 Å for the 150 line/mm grating) that's probably good enough.

ii. *reidentify* uses the output of *identify* as a first guess on a different lamp. You don't need a lampref for every objectname, usually about 3 for slits on the left, center, and right areas of the slitmask.

(b) Making a sensitivity function (do this once per observing run)

i. You really want to observe the spectrophotometric standard star both with and without a dichroic filter so that 2nd order light doesn't ruin the red half of your sensitivity function.

ii. We must process the spectrum of the spectrophotometric standard star analogously to how we have processed the multislit spectra in *bogus*. Run *nglccdproc* on spectroscopic image(s) of spectrophotometric standard star and on the flat that you will use. Then use *response* to normalize the flat so that only pixel-to-pixel variations are left. Now use *refspec* and *dispcor* to wavelength calibrate each image. You need to start by making a wavelength solution to a longslit arc spectrum as described for multislit spectra above.

iii. It's best to check linear vs. non-linear in *dispcor*, but it seems to make no difference. *flux=no* because you want to conserve flux **density**.

iv. In *noao.onedspec* you will find the tasks *standard* and *sensfunc* which you will now use to make the sensitivity function. *epar standard* and input the name of your spectrophotometric standard star. *standard* reads in a one-dimensional wavelength-calibrated spectrum, averages that spectrum into ~50 Å bins, and then writes out a file listing your counts in those bins and the calibrated flux in those bins. To be careful, you should use the extinction correction here and when *bogus2* calls *calibrate* below in order to properly account for a difference in airmass between when the standard star was measured and the calibrated spectrum and for a difference in airmass between standard star and science observations. *extinct=yes file=keckextinction.dat*. This doesn't seem to make a big difference, though. *sensfunc* then reads in this output file (there can be multiple if you took multiple observations) and determines the average sensitivity function to transform counts into flux. You should control the choice of function and order interactively; *func=spline3* is probably necessary to get a good fit. Note that it is very important to have the right exposure time listed in the header for each standard star frame and to have the gain in this image be the same as in everything you will apply the sensitivity function to (should be 1).

   v. There's this cool routine called *sbands* that you can use to figure out what the photometry of your standard star should be and thereby cross-check this sensitivity function.

(c) Running *bogus2* (once per object)

   i. *epar bogus2* and put in the desired objectname and proper lampref for wavelength calibration. *epar calibrate* to make sure that *fnu=yes* since we use fnu units for galaxies (but f_lambda i.e. *fnu=no* for quasars) and *extinct=yes* to handle airmass differences.

   ii. Run *bogus2* to wavelength calibrate and flux calibrate. This uses *refspec* and *dispcor* to refer the spectrum to a wavelength solution and then output the .w (wavelength calibrated) spectrum. Then it uses *calibrate* to create the .f (flux calibrated) spectrum. *bogus2* asks the user to mark skylines at 5577, 5893, 6300, 7276, 7751, 8344, and 8827 Å using *k k* in splot (*w e e* can be used to zoom) and writes the apparent wavelengths to **splot.log**. The first 3 of those skylines are bright (5893 is blended for 150 line/mm and looks like 1 line) the next 3 are the 2nd of each set of bright bandheads, and the 7th is the 2nd line of the next bright bandhead after a solitary O line. Hit q when done. **splot.log** should have 2 blank lines, a header, and 7 data lines. The user is then asked to run a fortran program called **lamoffset150** (you can call this with *!lamoffset150*) that reads **splot.log** and calculates the pixel shift for each of these skylines and the average pixel shift. If there is more than a few Å dispersion in these shifts then it is time to make a new lampref.

   iii. *bogus2* gives warnings but if it runs it is probably working fine. To re-run on the same object you'll need to delete **objectname.lamp_s.ms.fits** and **database/idobjectname.lamp_s.ms**.

   iv. I have modified *bogus2* to also flux-calibrate the **objectname.sky_s.ms.fits** spectrum and to copy its 4th band into a file called **objectname_sigma.f.fits** and an ascii file called **objectname_sigma.dat**. If *apall* is set to multispec format then this 4th band represents the sigma spectrum of the version for which *bogus* did not sky subtraction. The sky subtraction was performed by *apall* using the exact same aperture and background region selected for the main *objectname* spectrum. This includes a proper representation of the errors introduced by the original sky levels, which is the best guess at the errors in our final spectrum since we didn't get rid of sky noise just sky average in running *background* during *bogus*.

   v. When you examine the flux- and wavelength-calibrated spectrum objectname.f in *splot*, *$* switches between wavelength and pixel number as x-coordinates.

V. Determine redshifts and uncertainties (once per object)

(a) **mslit_fndz**. Set up input file **mslit_fndz.inp** properly to be in a subdirectory **Input** of the directory from which you call **mslit_fndz**.

(b) Options are found by typing ? and pop up if you mistype. PGPLOT hangs pretty easily so be careful. You can overplot the sigma spectrum, hit *e e* to zoom, hit *S* to smooth by 2, hit *s* to undo smoothing, hit *w* to determine wavelength, and hit *f* to put Lyman alpha wavelength there and see the prediction for other relevant absorption lines.

(c) If you end up smoothing the final spectrum by n pixels, you should then *imarith objectname_sigma.f* / $\sqrt{n}$ *objectname_sigma_smoothed.f* to represent the noise level in the smoothed spectrum. Although smoothing makes neighboring pixels highly correlated, this reduced sigma spectrum will properly represent the uncertainty in a given n-pixel wide bin of the smoothed spectrum, and will look right visually.