Gemini Cookbook V2.0

**Changes:**

**1.01 Added How to retrieve data from Gemini Science Archive:**

**2.0 Updated for new archive. Automated python code put in. Uploaded in Google Drive folder.**

**3.0.0 Reworking for ZTF using new Gemini chip. Work in progress.**

This manual contains a step-by-step guide on how to reduce gemini GMOS spectra, especially that collected by iPTF.

The original version was prepared by Christoffer Fremling and Emir Karamehmetoglu, based on a workshop given by Brad Cenko. Since then, maintained and updated by Emir Karamehmetoglu.

**Section I. Checklist Before Beginning**

**Required software:**

Ureka: ( for iraf, pyraf, Gemini pipeline, aparr, splot)

NOTE: Ignore their sales pitch for the new python environment, just download Ureka.

<http://ssb.stsci.edu/ureka/>

LACos\_Spec (Cosmic ray rejection—optional)

<http://www.astro.yale.edu/dokkum/lacosmic/lacos_spec.cl>

And installation instructions for LACos\_spec here:

<http://www.astro.wisc.edu/~cigan/reducing/lacos/lacos.html>

—Someway to interact with fits headers:

Ureka python installation above comes with standard tools for interacting with fits headers.

DS9 works, too. We also have a matlab & python script that you can use.

**Data:**

In this cookbook we work through a standard iPTF example, where we have:

2 science images, (1 in each grating),

2 standard star images, (1 in each grating)

4 arc images, (1 for each science image and standard star in each grating.)

4 flat images, (1 for each science image and standard star in each grating.)

#Here is a series of 2 python scripts to run this whole reduction for you:

[Readsort.py](http://drive.google.com/open?id=0B8UvncxJsb0cdVE1MGxaM0lVVlU)

[Trialreduce.py](http://drive.google.com/open?id=0B8UvncxJsb0cTHZyRjB3T255eEk)

**Python reduction script:**

In a new terminal window, switch to Ureka python by typing *ur\_setup*. Navigate to the folder where you have the data along with the scripts readsort.py and trialreduce.py.

First run readsort.py, by typing:

*python readsort.py*

Then run trialreduce.py by typing:

*python trialreduce.py*

Trialreduce.py will go through all of the steps listed in this manual, but automatically instead, by calling the relevant iraf tasks. You will need to manually select the extraction region and the fit to the trace in iraf.apall ([see the the relevant section for instructions.)](https://docs.google.com/document/d/1rWVIlJamvizdmd770xn_nHQtyZmA1oX9CzkYZ80FwxQ/edit#heading=h.vj8vabeyunlc) You will need to select the standard star box regions using ‘a’ key, to fit a sensitivity curve to the standard star during flux calibration ([instructions](https://docs.google.com/document/d/1rWVIlJamvizdmd770xn_nHQtyZmA1oX9CzkYZ80FwxQ/edit#heading=h.iopg7ls5ce3n).) You will need to enter the printed airmass and exposure time for each SN in the flux calibration step, due to a bug in IRAF. These values will be printed to terminal immediately before IRAF asks for them. Finally, you will get to inspect the final flux calibrated and combined spectrum with splot.

The script writes out .ascii files for both the combined spectrum, and the red and blue parts individually (with no headers), so that you can run SNID on them. It also saves all intermediate files, though you can probably delete them if you like the output and have uploaded the spectrum to the marshal. To see these outputs, type:

*ls \*.ascii*

on the the terminal to see them.

#Here is an alternative matlab script written by Christoffer which goes through this entire cookbook automatically as well.

[gemini\_script.m](http://drive.google.com/open?id=0B8UvncxJsb0cTGFHN1V4U3IwMFk)

#If your reduction is unique, and you would rather copy paste pieces of code 1 step at a time, that starts in [section II](https://docs.google.com/document/d/1rWVIlJamvizdmd770xn_nHQtyZmA1oX9CzkYZ80FwxQ/edit#heading=h.5o7b4oy6nyxg). This is useful for when you need more fine control over some steps, if the spectrum is difficult.

#For doing the above step-by-step, you still need the following script:

[readsort.py](http://drive.google.com/open?id=0B8UvncxJsb0cdVE1MGxaM0lVVlU)

**Data Retrieval:**

You will need to make an account on the Gemini archive:

[http://archive.gemini.edu](http://archive.gemini.edu/)

Note: The outdated instructions below were written for the previous archive. However, they remain largely similar/the same. Here I give a new overview, but you may look at the outdated section for more details on any individual step, due to the similarities.

Make an account on the Gemini archive, register yourself as PI for the iPTF proposal that is currently active (to access data from that proposal). Then, when new data arrives, simply search in Gemini N or S for the data from that day, and take the data products required as listed above in the previous section. The standard setup is a a set of science and comparison star spectra, in r400 and b600 filters, and a spectral flat and arc for each of the science and comparison star spectra. The comparison star is observed immediately before or immediately after the iPTF trigger. Simply mark all of those from the day, (taking care NOT to include the acquisition images), and press download. Then, place the fits images in that download, uncompressed, in the same folder as the above python script. You may use the following command the uncompress the images:

*gunzip \*bz2*

**Outdated Instructions:**

~~You will need to make an account at the CADC:~~

[~~http://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/en/gsa/~~](http://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/en/gsa/)

~~After you have your account, you now have access to all the public data. The gemini data are in the Gemini science archive, reachable at:~~

[~~http://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/en/gsa/~~](http://www.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/en/gsa/)

For iPTF data, you will need to put yourself as a PI in the current Gemini program using the name of the program and the PhaseII proposal password. At the list on the left, under the “Proprietary Data Access”, click on “Access your PI Data”. Enter the Gemini Program ID and Phase II password, (ask Emir if you are not sure). Refresh. From then on, you may access processed data for the program under this tab, by selecting the program name, and pressing on “List datasets”. Processed data puts together all of the calibration and science images for you.

For rapid access to data which has not been put together for you automatically yet, you will need to either search for the object by its “Gemini name”, such as iPTF15acr, or search specifically that entire day’s observations by giving the data in the advanced search options. (The latter is necessary when there is a mistake with the name of the target in the OB.)

Download these datasets by selecting the “url list” option when prompted in a pop-up, and using wget with your CADC account credentials under html mode:

wget --http-user=<cadc username> --http-password=<cadcpassword> --content-disposition -i cadcUrlList.txt

Where cadcUrlList.txt is the standard name of the downloaded file.

Additionally during rapid access, the calibration files must be downloaded separetly. After searching and finding your target, scroll to the right and press on “Calibration Files”. From here, select all the calibration files that interest you. Normally, these are 2 standard star spectra in 2 separate grisms, 4 arc spectra, 4 flats. You may also download the acquisitions images from here.

The data are available on this server almost immediately after being observed, (within the hour). So it is possible to refresh the calibration files list while waiting for it all to arrive, if they observe the standard after your target.

**Section II. Reduction Using Gemini-Specific Pipeline**

Comments are denoted with a leading #, while commands to be typed are usually placed immediately after a comment summarizing what it does.

#Organize the fits files by running:

python readsort.py

#Start ureka

ur\_setup

#start pyraf

pyraf

#go to gemini/gmos

gemini

gmos

**Producing Normalized Flats:**

The gsflat fits a polynomial to each CCD. Check the order of the fit to ensure that it is not overfitting the small scale variations, but only to the overall trend. Press q if you are satisfied with each fit, to move on to the next. If you would like to change the order of the fit type:

“ : o X” Where X is an integer number greater than 0. (The dashed line is the fit, not the solid line).

#Produce normalized flats

gsflat inflats=@flatb600\_1.txt specflat=b600\_norm\_flat\_1.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatr400\_1.txt specflat=r400\_norm\_flat\_1.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_2.txt specflat=b600\_norm\_flat\_2.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatr400\_2.txt specflat=r400\_norm\_flat\_2.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

#ZTF dithered obs:

gsflat inflats=@flatb600\_520\_1.txt specflat=b600\_520\_norm\_flat\_1.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_520\_2.txt specflat=b600\_520\_norm\_flat\_2.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_520\_3.txt specflat=b600\_520\_norm\_flat\_3.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_520\_4.txt specflat=b600\_520\_norm\_flat\_4.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_525\_1.txt specflat=b600\_525\_norm\_flat\_1.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_525\_2.txt specflat=b600\_525\_norm\_flat\_2.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_525\_1.txt specflat=b600\_525\_norm\_flat\_1.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_525\_2.txt specflat=b600\_525\_norm\_flat\_2.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_520\_3.txt specflat=b600\_520\_norm\_flat\_3.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

#Standard star

#with bias

gsflat inflats=@flatb600\_520\_1.txt specflat=b600\_520\_norm\_flat\_1.fits bias=MCbias.fits fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_520\_2.txt specflat=b600\_520\_norm\_flat\_2.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_525\_1.txt specflat=b600\_525\_norm\_flat\_1.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_525\_2.txt specflat=b600\_525\_norm\_flat\_2.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

gsflat inflats=@flatb600\_520\_3.txt specflat=b600\_520\_norm\_flat\_3.fits fl\_bias=no fl\_inter=no fl\_detec=yes fl\_seprows=no

**Reducing the Science, Arc, and Standard Star Images:**

We will now apply the normalized flats, as well as use the overscan region to do bias subtraction on our images with the command gsreduce. If you ran the python script the organize the files, the below commands should work automatically. However, the inputs are self-explanatory and can be manually subtracted. This part of the script requires no user interaction, it will run to the end automatically and produce the reduced images by appending “gs” to the beginning of each fits image.

#Reduce Science, Arc, Standard Star Images

gsreduce inimages=@object\_r400\_1.txt flatim=r400\_norm\_flat\_1.fits fl\_bias=no

gsreduce inimages=@CuAr\_r400\_1.txt flatim=r400\_norm\_flat\_1.fits fl\_bias=no

gsreduce inimages=@object\_b600\_1.txt flatim=b600\_norm\_flat\_1.fits fl\_bias=no

gsreduce inimages=@CuAr\_b600\_1.txt flatim=b600\_norm\_flat\_1.fits fl\_bias=no

gsreduce inimages=@object\_r400\_2.txt flatim=r400\_norm\_flat\_2.fits fl\_bias=no

gsreduce inimages=@object\_b600\_2.txt flatim=b600\_norm\_flat\_2.fits fl\_bias=no

gsreduce inimages=@CuAr\_r400\_2.txt flatim=r400\_norm\_flat\_2.fits fl\_bias=no

gsreduce inimages=@CuAr\_b600\_2.txt flatim=b600\_norm\_flat\_2.fits fl\_bias=no

Important note:

WE ASSUME 1 FLAT FOR EACH OBJ/STANDARD STAR. IF NOT, REMOVE EXTRA OBSERVATIONS FROM DIRECTORY

#ZTF:

#Arc:

gsreduce inimages=@CuAr\_b600\_520\_1.txt flatim=b600\_520\_norm\_flat\_1.fits fl\_bias=no

gsreduce inimages=@CuAr\_b600\_525\_1.txt flatim=b600\_525\_norm\_flat\_1.fits fl\_bias=no

#####object & std in 520  #####object in 525

gsreduce inimages=@CuAr\_b600\_520\_1.txt flatim=b600\_520\_norm\_flat\_1.fits fl\_bias=no

gsreduce inimages=@CuAr\_b600\_525\_1.txt flatim=b600\_525\_norm\_flat\_1.fits fl\_bias=no

gsreduce inimages=@object\_b600\_520\_1.txt flatim=b600\_520\_norm\_flat\_1.fits fl\_bias=no

gsreduce inimages=@object\_b600\_520\_2.txt flatim=b600\_520\_norm\_flat\_2.fits fl\_bias=no

gsreduce inimages=@object\_b600\_520\_3.txt flatim=b600\_520\_norm\_flat\_3.fits fl\_bias=no

gsreduce inimages=@object\_b600\_520\_4.txt flatim=b600\_520\_norm\_flat\_4.fits fl\_bias=no

gsreduce inimages=@object\_b600\_525\_1.txt flatim=b600\_525\_norm\_flat\_1.fits fl\_bias=no

gsreduce inimages=@object\_b600\_525\_2.txt flatim=b600\_525\_norm\_flat\_2.fits fl\_bias=no

#standard star x2

gsreduce inimages=@object\_b600\_525\_1.txt flatim=b600\_525\_norm\_flat\_1.fits fl\_bias=no

gsreduce inimages=@object\_b600\_525\_2.txt flatim=b600\_525\_norm\_flat\_2.fits fl\_bias=no

#alternative to flats for arcs

gsreduce inimages=@CuAr\_b600\_520\_1.txt fl\_flat=no fl\_bias=no

gsreduce inimages=@CuAr\_b600\_520\_2.txt fl\_flat=no fl\_bias=no

gsreduce inimages=@CuAr\_b600\_525\_1.txt fl\_flat=no fl\_bias=no

gsreduce inimages=@objects\_520.txt bias=MCbias flatim=b600\_520\_norm\_masterflat fl\_fixpix=no fl\_oversize=no, fl\_vardq=yes fl\_fulldq=yes

**2D Wavelength calibration of Arc images:**

We will calculate a wavelength solution to our arc spectra. The program gswavelength will suggest a wavelength solution, (which is usually quite good.) You can inspect the fit by pressing f, then remove outliers via standard IRAF commands. (d for delete.) When prompted, you should write NO to automatically fit to everything; however, watch out for the RMS and Zshift columns. We find that Z-shift greater than or equal to 10-4 suggests trouble for the automatic fitting. You may need to go in and adjust individual fits to reduce the RMS scatter as well.

##Wavelength calibrate ARCs in 2d

gswavelength inimages=gs//@CuAr\_r400\_1.txt

gswavelength inimages=gs//@CuAr\_b600\_1.txt

gswavelength inimages=gs//@CuAr\_r400\_2.txt

gswavelength inimages=gs//@CuAr\_b600\_2.txt

#ZTF

gswavelength inimages=gstest//@CuAr\_b600\_520\_1.txt

gswavelength inimages=gstest//@CuAr\_b600\_525\_1.txt

Now we apply the wavelength solution which we calculated in the previous step, to our science and standard-star images. This part should execute without any input from the user.

**Applying Wavelength Calibration to Science Images**

##Apply transform

gstransform inimages=gs//@object\_r400\_1.txt wavtran=gs//@CuAr\_r400\_1.txt

gstransform inimages=gs//@object\_b600\_1.txt wavtran=gs//@CuAr\_b600\_1.txt

gstransform inimages=gs//@object\_b600\_2.txt wavtran=gs//@CuAr\_b600\_2.txt

gstransform inimages=gs//@object\_r400\_2.txt wavtran=gs//@CuAr\_r400\_2.txt

#ZTF:

gstransform inimages=gs//@object\_b600\_520\_1.txt wavtran=gs//@CuAr\_b600\_520\_1.txt

gstransform inimages=gs//@object\_b600\_520\_2.txt wavtran=gs//@CuAr\_b600\_520\_1.txt

gstransform inimages=gs//@object\_b600\_520\_3.txt wavtran=gs//@CuAr\_b600\_520\_1.txt

gstransform inimages=gs//@object\_b600\_520\_4.txt wavtran=gs//@CuAr\_b600\_520\_1.txt

gstransform inimages=gs//@object\_b600\_525\_1.txt wavtran=gs//@CuAr\_B600\_525\_1.txt

gstransform inimages=gs//@object\_b600\_525\_2.txt wavtran=gs//@CuAr\_B600\_525\_1.txt

gstransform inimages=gs//@object\_b600\_1.txt wavtran=gs//@CuAr\_b600\_1.txt

gstransform inimages=gs//@object\_b600\_2.txt wavtran=gs//@CuAr\_b600\_2.txt

gstransform inimages=gs//@object\_r400\_2.txt wavtran=gs//@CuAr\_r400\_2.txt

At this point, we are done using the Gemini specific tools. These images should now fit into any other standard pipeline or method of spectral reduction. In Section II, we will continue with cosmic rejection, and extraction of a flux calibrated spectrum using methods recommended by Brad Cenko.

**Section III. Producing a First-Look 1D Spectrum**

We will use iraf imcopy command to create copies of the properly reduced images to work on, in order to not repeat any of the previous steps if we get errors. You may do this in many different ways. Your reduced images will start with tgs, and their science data content is stored in [SCI].

You should now use “gethead” or DS9 to check the object names:

#spit out object names to terminal from within pyraf using gethead

!gethead tgsN\*.fits -x 0 OBJECT GRATING CENTWAVE

N.B: It is difficult to standardize the names and use of these commands without a script to easily keep track of filenames as we go under the same variables/lists. Thus here, the python script can be your best friend or just manual header checking!  We are not providing a second script to sort the files again.

To do it semi-automatically, type the first command below, then use the output to copy and paste this into the <paste output here line by line>. The outputs are now saved in ptfsnX.fits for the science images, and ptfstarX.fits for standard stars, where X is the filenumber. You should check that the files are named correctly by checking the fits headers, with gethead, DS9, or your own preferred method!! Check the “OBJECT” keyword.

##Create Copies of Final Products to Work on

files \*tgs\* | sed 's/$/[SCI]/'

imcopy <paste output here line by line> ptfsn1.fits

imcopy <paste output here line by line> ptfstar1.fits

imcopy <paste output here line by line> ptfsn2.fits

imcopy <paste output here line by line> ptfstar2.fits

**Cosmic Ray Rejection with LACos\_spec**

If you properly followed the installation instruction, LACos\_spec should be available to you. However, you may need to run “stsdas” if you are not putting it in to your login.cl file before lacos\_spec is available.

#Go to STSDAS to have access to LACos\_spec

stsdas

We will use a standard of 3 iterations to remove cosmics from our science images. This should **not** be run on standard stars!! You can copy and paste the command below, or alternatively, you can adjust your default parameters for once with:

“epar lacos\_spec”,

save, and run lacos\_spec by simply typing:

“lacos spec <filename> <filename>.cr <filename>.crmask”,

where you substitute the name of your file for filename.

#Remove Cosmics w/ LACos\_spec.

lacos\_spec input=ptfsn1 output=ptfsn1.cr outmask=ptfsn1.crmask gain=1.0 readn=3.85 xorder=9 yorder=3 sigclip=4.5 sigfrac=0.5 objlim=1.0 niter=3 verbose=yes mode=al

lacos\_spec input=ptfsn2 output=ptfsn2.cr outmask=ptfsn2.crmask gain=1.0 readn=3.85 xorder=9 yorder=3 sigclip=4.5 sigfrac=0.5 objlim=1.0 niter=3 verbose=yes mode=al

#Run with parameters already set with epar

lacos\_spec ptfsn1 ptfsn1.cr ptfsn1.crmask

lacos\_spec ptfsn2 ptfsn2.cr ptfsn2.crmask

#ZTF:

lacos\_spec input=ptfsn1 output=ptfsn1\_f.cr outmask=ptfsn1\_f.crmask gain=1.65 readn=4.06 xorder=9 yorder=3 sigclip=4.5 sigfrac=0.5 objlim=1.0 niter=3 verbose=yes mode=al

lacos\_spec input=ptfsn2 output=ptfsn2\_f.cr outmask=ptfsn2\_f.crmask gain=1.65 readn=4.06 xorder=9 yorder=3 sigclip=4.5 sigfrac=0.5 objlim=1.0 niter=3 verbose=yes mode=al

lacos\_spec input=ptfsn3 output=ptfsn3\_f.cr outmask=ptfsn3\_f.crmask gain=1.65 readn=4.06 xorder=9 yorder=3 sigclip=4.5 sigfrac=0.5 objlim=1.0 niter=3 verbose=yes mode=al

lacos\_spec input=ptfsn4 output=ptfsn4\_f.cr outmask=ptfsn4\_f.crmask gain=1.65 readn=4.06 xorder=9 yorder=3 sigclip=4.5 sigfrac=0.5 objlim=1.0 niter=3 verbose=yes mode=al

lacos\_spec input=ptfstar1 output=ptfstar1\_f.cr outmask=ptfstar1\_f.crmask gain=1.65 readn=4.06 xorder=9 yorder=3 sigclip=4.5 sigfrac=0.5 objlim=1.0 niter=3 verbose=yes mode=al

lacos\_spec input=ptfstar2 output=ptfstar2\_f.cr outmask=ptfstar2\_f.crmask gain=1.65 readn=4.06 xorder=9 yorder=3 sigclip=4.5 sigfrac=0.5 objlim=1.0 niter=3 verbose=yes mode=al

**Using IRAF apall to Obtain a Simple Spectrum**

We will use apall to fit the trace of the supernova and extract the 1D spectrum. This step will also subtract out the background. Make sure you fit the trace to either the host or the supernovae depending on what you are after. Check the reference image to get a simple example.

For apall, use “m” to mark the aperture, use “l” and “u” to set the limits and “d” to delete. You should try to capture as much of the supernova as possible. Also press “b” to check the background, and adjust the order as before using “: o X” where X is the order of the line to be fit. Press “f” to fit to the trace, which will fit a line to the maximum values, and adjust order of the fit as before. This will be aperture from which your spectrum will be extracted.

<Insert Trace Image Link Here>

#Run apall to extract 1d spectrum of science and standard-star images

apall input=ptfsn1.cr out=ptfsn1.ms nfind=1 interactive=yes find=yes recenter=yes resize=yes edit=yes trace=yes fittrace=yes t\_nsum = 50 t\_nlost = 50 extract=yes extras=yes review=yes background=fit

apall ptfstar1.fits out=ptfstar1.ms

apall input=ptfsn2.cr out=ptfsn2.ms nfind=1 interactive=yes find=yes recenter=yes resize=yes edit=yes trace=yes fittrace=yes t\_nsum = 50 t\_nlost = 50 extract=yes extras=yes review=yes background=fit

apall ptfstar2.fits out=ptfstar2.ms

The .ms files contain your spectra.

#ZTF:

apall input=ptfsn1\_f.cr out=ptfsn1\_f.ms nfind=1 interactive=yes find=yes recenter=yes resize=yes edit=yes trace=yes fittrace=yes t\_nsum = 50 t\_nlost = 50 extract=yes extras=yes review=yes background=fit

apall input=ptfsn2\_f.cr out=ptfsn2\_f.ms nfind=1 interactive=yes find=yes recenter=yes resize=yes edit=yes trace=yes fittrace=yes t\_nsum = 50 t\_nlost = 50 extract=yes extras=yes review=yes background=fit

apall input=ptfsn3\_f.cr out=ptfsn3\_f.ms nfind=1 interactive=yes find=yes recenter=yes resize=yes edit=yes trace=yes fittrace=yes t\_nsum = 50 t\_nlost = 50 extract=yes extras=yes review=yes background=fit

apall input=ptfsn4\_f.cr out=ptfsn4\_f.ms nfind=1 interactive=yes find=yes recenter=yes resize=yes edit=yes trace=yes fittrace=yes t\_nsum=50 t\_nlost=50 extract=yes extras=yes review=yes background=fit

apall input=ptfstar1\_f.cr out=ptfstar1\_f.ms nfind=1 interactive=yes find=yes recenter=yes resize=yes edit=yes trace=yes fittrace=yes t\_nsum = 50 t\_nlost = 50 extract=yes extras=yes review=yes background=fit

apall input=ptfstar2\_f.cr out=ptfstar2\_f.ms nfind=1 interactive=yes find=yes recenter=yes resize=yes edit=yes trace=yes fittrace=yes t\_nsum = 50 t\_nlost = 50 extract=yes extras=yes review=yes background=fit

apall input=ptfsn1\_f.cr out=ptfsn1\_f.ms nfind=1 interactive=yes find=yes recenter=yes resize=yes edit=yes trace=yes fittrace=yes t\_nsum = 50 t\_nlost = 50 extract=yes extras=yes review=yes background=fit

apall input=ptfsn2.cr out=ptfsn2.ms nfind=1 interactive=yes find=yes recenter=yes resize=yes edit=yes trace=yes fittrace=yes t\_nsum = 50 t\_nlost = 50 extract=yes extras=yes review=yes background=fit

apall ptfstar1.fits out=ptfstar1.ms

apall input=ptfsn3.cr out=ptfsn3.ms nfind=1 interactive=yes find=yes recenter=yes resize=yes edit=yes trace=yes fittrace=yes t\_nsum = 50 t\_nlost = 50 extract=yes extras=yes review=yes background=fit

apall input=ptfsn4.cr out=ptfsn4.ms nfind=1 interactive=yes find=yes recenter=yes resize=yes edit=yes trace=yes fittrace=yes t\_nsum = 50 t\_nlost = 50 extract=yes extras=yes review=yes background=fit

apall ptfstar2.fits out=ptfstar2.ms

**Section IV. Producing Flux Calibrated Spectra**

At this point, you have your 1D spectra, and can use a variety of methods to obtain your final spectrum. The matlab script uses the same principles but does this part within matlab. You may also use tools provided in gemini/gmos.

(For example see this link: <http://eduard.hles.nl/thesis/geminiReductions.shtml>)

We outline one possible method.

**Flux Calibration with IRAF**

**Identify Regions to Fit a Sensitivity Function**

We elect to skip telluric correction. We fit a sensitivity function to the continuum of the standard star by first selecting the regions to fit. (Do not select the regions with tellurics.) You will need the name of your standard star from the OBJECT header which has been pulled out twice previously. Also you will need the airmass and exposure time.

Finally you will need an extinction file for your star and to have a directory containing calibration data for standard stars. These can be acquired from the usual sources, (or just through google.)

#To search if iraf already has your standard star, type:

page onedstds$README

Copy the location, (such as “onedstds$iidscal/”), this is your path to calibration directory.

#This will print the relevant values for all files to your screen

!gethead tgsN\*.fits -x 0 OBJECT AIRMASS EXPTIME

We use iraf program standard for identifying regions to fit a sensitivity function to. Delete regions that you do NOT want to fit with “d”.

#Identify regions where we want to fit sensitivity function

standard input=ptfstar1.ms output=stdstar1 star\_name=<STD NAME> airmass=<INSERT AIRMASS> exptime=<INSERT EXPOSURE TIME IN DECIMAL SECONDS> answer=yes extinction=<PATH TO EXTINCTION FILE> caldir=<PATH TO CALIBRATION DIRECTORY>  observatory=Keck

standard input=ptfstar2.ms output=stdstar2 star\_name=<STD NAME> airmass=<INSERT AIRMASS> exptime=<INSERT EXPOSURE TIME IN DECIMAL SECONDS> answer=yes extinction=<PATH TO EXTINCTION FILE> caldir=<PATH TO CALIBRATION DIRECTORY> observatory=Keck

**Fit Sensitivity Function**

#Fit a sensitivity function to previously identified regions

sensfunc standards=stdstar1 sensitivity=sensstar1 observatory=Keck order=4

sensfunc standards=stdstar2 sensitivity=sensstar2 observatory=Keck order=4

**Calibration**

Use IRAF “calibrate” to calibrate the data.(extinction probably is = onedstds$kpnoextinct.dat)

Bug with IRAF, you may need to manually provide exposure time and airmass of sn.

onedspec.calibrate input=ptfsn1.ms out=ptfsn1.f extinction=onedstds$kpnoextinct.dat observatory=Keck ignoreaps=yes sensitivity=sensstar1.fits

#ENTER ONE AT A TIME

onedspec.calibrate input=ptfsn2.ms out=ptfsn2.f extinction=onedstds$kpnoextinct.dat observatory=Keck ignoreaps=yes sensitivity=sensstar2.fits

Combine Two Spectra if Using Different Filters

We use scombine to stitch the spectra together. Note that your sample region can differ based on your spectra! Change the sample parameter to account for this.

scombine input=ptfsn1.f,ptfsn2.f out=finalspectrum.ms reject=avsigclip scale=median sample=5500:6500

**Possible Improvements**

While there are presumably many, here are a few major ones.

1) Make sure the inspect your wavelength calibration and really understand it to get better spectra.

2) You can apply a telluric correction

3) Fitting the trace is quite barbaric in apall, and could be detracting from the S/N. If you need higher S/N, you may want to use something that calculates a profile for your source.

4) Your Contributions

Troubleshooting

Section Under Development!

-When in doubt, type “epar <module>” to get a GUI version of the modules, and mess with the parameters there. The help button can also be useful!

-If you get an error in IRAF, stuff in your directory could be corrupted, and you may have to start from a previous step or clear everything!

-IRAF won’t like you accessing the fits file while it’s working on it.

-If apall crashes due to bad parameters from the matlab script, try running pyraf again. This sometimes fixes the issue with faulty parameters or temp files.

-IRAF will give an error if it doesn’t find the trace correctly. You will have to keep messing with apall, being careful with the order you choose, the background you define, etc. until IRAF doesn’t complain. Otherwise you cannot move on to produce a final spectrum. We set a generous limit, so this shouldn’t be a huge problem.

-In flux calibration, we use onedspec.calibrate, NOT longslit.calibrate. Sometimes Iraf will default to the latter, which does not take into account airmass or exposure time. This can cause a conflict. The fix is to explicitly call onedspec.calibrate instead of just calibrate.

-If the Iraf function “standard” has too few points identifying the continuum, you may force it to find a better automatic fit by just typing “standard” and providing the parameters manually. Not sure why this works, but it does. It will then overidentify by quite a lot. Alternatively, select continuum regions with “a”.

-When fitting for the wavelength solution, if gsreduce fails, (due to r150 grism or other), increase the “match” from -13, -100 or so. This sometimes fixes the problem. Go through the line identifications to make sure they are not off.

-if using wspectext, you might need to use the formalism wspectext foo.fits[\*,1]

The work may be used, edited, altered, distributed in part or whole by anyone, for any purpose, and for any duration, as long as credit is given to the original preparers Emir Karamehmetoglu and Christoffer Fremling. CC by attribution 2014.