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.**

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

**3.1.0 Minimal working example for ZTF is now live. Re-written to use Astroconda as Ureka is completely unsupported now. Also put on to Github.**

This manual contains a step-by-step guide on how to reduce gemini GMOS spectra, especially that collected by ZTF. It is maintained by Emir Karamehmetoglu. The reductions were prepared based on a workshop given by Brad Cenko and implemented by Christoffer Fremling and Emir Karamehmetoglu.

**Section I. Checklist Before Beginning**

**Required software:**

**task lacos\_spec=/Users/emir/iraf/lacos\_spec.cl**

**For ZTF:**

Set up an Astroconda virtual environment with iraf using python version 2.7. Follow the instructions in their readthedocs.

<https://astroconda.readthedocs.io/en/latest/getting_started.html>

Also look here for Gemini’s own instructions (though they are the same). This link also has a data reduction cookbook for Gemini GMOS data though we do things a bit differently for now.

<http://ast.noao.edu/sites/default/files/GMOS_Cookbook/GettingStarted.html>

Get LACos spec and add it to your login.cl using the following command:

task lacos\_spec= /path/to/iraf\_home/lacos\_spec.cl

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:

DS9 works… anything really. We also have python script that you can use.

**Data:**

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

4 science images, (2 in each central wavelength (centwave) of the grating),

2 standard star images, (1 in each centwave)

2 arc images, (1 for each centwave).

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

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

Readsort.py

Ztf\_reduce.py

**Python reduction script:**

In a new terminal window, activate the astroconda virtual environment by typing “source activate <envname> (in the instructions above it should be iraf27). Navigate to the folder where you have the data along with the scripts readsort.py and trialreduce.py. Here type mkiraf and then add the task lacos\_spec into your login.cl file, under the other tasks. Then

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’s 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).) Finally, you will get to inspect the final flux calibrated and combined spectrum with splot. All the intermediate products are saved and can be checked or any step can be re-run in your own python script or inside of pyraf.

The script writes out .ascii files for the combined spectrum with no headers, so that you can run SNID on them. To see these outputs, type:

*ls \*.ascii*

on the the terminal.

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/)

Make an account on the Gemini archive, register yourself as PI for the relevant 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 set of science and comparison star spectra, in b600 filters with centwave of 520 and 525, and a spectral flat for each of the science and comparison star spectra. Take the arc spectra with correct grism/grating/centwave as well (these will be taken at the end of the day but in a pinch use an older one). The comparison star is usually observed immediately before or immediately after the trigger, or whenever possible. 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 to uncompress the images:

*gunzip \*bz2*

**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.

#Activate astroconda

Source activate <env-name> (Such as iraf27)

#Organize the fits files by running:

python readsort.py

#start pyraf

pyraf

#go to gemini/gmos and also load stsdas

gemini

gmos

stsdas

**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).

#ZTF dithered obs:

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

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

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

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

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

gsflat inflats=@flatb600\_525\_3.txt specflat=b600\_525\_norm\_flat\_3.fits fl\_bias=no fl\_inter=yes 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.

# fl\_crspec=yes will only work if lacos\_spec is installed. Otherwise set that to no. Make sure #you don’t turn it on for the arc spectrum.

#####object & std in 520/525  #####

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 fl\_crspec=yes

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

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

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

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

gsreduce inimages=@object\_b600\_525\_3.txt flatim=b600\_525\_norm\_flat\_3.fits fl\_bias=no

fl\_crspec=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\_b600\_525\_1.txt

gswavelength inimages=gs//@CuAr\_b600\_520\_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**

#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\_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\_525\_3.txt wavtran=gs//@CuAr\_B600\_525\_1.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 extraction of a flux calibrated spectrum using standard IRAF.

**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!

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 ztfsnX.fits for the science images, and ztfstarX.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> ztfsn\_520\_1.fits

imcopy <paste output here line by line> ztfsn\_520\_2.fits

…

imcopy <paste output here line by line> ztfstar\_520.fits

etc.

**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. **Replace the input parameter with the name of the spectrum.** The object is typically located at around a pixel of y~1050 or so.

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.

The .ms files contain your spectra.

#ZTF:

apall input=**ztfsn\_520\_1.fits** out=**ztfsn\_520\_1.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

**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. 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. 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= ztfstar\_520.ms output=stdstar\_520 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= ztfstar\_525.ms output=stdstar\_525 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=stdstar\_520 sensitivity=sensstar520 observatory=Keck order=4

sensfunc standards=stdstar\_525 sensitivity=sensstar525 observatory=Keck order=4

(Might need higher order, see the fit.)

**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=ztfsn\_520\_1.ms out= ztfsn\_520\_1.f extinction=onedstds$kpnoextinct.dat observatory=Keck ignoreaps=yes sensitivity=sensstar520.fits

#Make sure to ENTER ONE AT A TIME, or the calibrate script bugs out.

Combine the spectra:

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=ztfsn\_520\_1.f,ztfsn\_525\_1.f,etc.. out=finalspectrum.ms reject=minmax scale=none nlow=0 nhigh=1 nkeep=2 sample=4500:6500

scombine input=ztfsn\_520\_1.f,ztfsn\_525\_1.f,etc.. out=finalspectrum.ms reject=avsigclip scale= sample=4500:6500

#combine in anyway you like, I have given two examples.

**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. Apply master flat and master bias to images

2) You can apply a telluric correction

3) Use optimal tracing instead of apall.

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.

-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 to -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, see the license for more info.