Flamingos-2 Data Reduction Cookbook

Release 1.1.1

Richard Shaw and Chris Simpson

2018-Mar-12

Contents

The Gemini Observatory[1] provides a variety of facility-class instruments for use by the professional astronomy community, including the FLAMINGOS-2 wide-field imager and multi-object spectrograph (sometimes abbreviated F2[2]), which is deployed on Gemini-South. This imaging spectrograph offers useful sensitivity in the near-infrared (NIR) from $0.95 - 2.4\mu m$, at high spatial resolution and low to moderate spectral resolution. FLAMINGOS-2 was constructed by the University of Florida Astronomy Department[3] and delivered in 2009 July; it was refurbished and commissioned in 2011-December. The F2 Status and Availability[4] page summarizes significant operations issues over the life of the instrument that may have affected the quality of the science data in the Archive.

Data from the commissioning periods to the present are offered in the Gemini Observatory Archive[5]. Raw data for this instrument is archived and made available for public use, but no calibrated, science-ready products are provided. This *Cookbook* is intended as a guide to data reduction and calibration for PIs and Archive users of data from FLAMINGOS-2. The descriptions, recipes, and scripts offered in this *Cookbook* will provide the necessary information for deriving scientifically viable (but not necessarily optimal) spectra and images from F2. However, users should be aware that the ultimate utility of the data for any specific scientific goal depends strongly upon a number of external factors, including:

- the environmental conditions that prevailed at the time of the observations (see the Gemini data quality assessment process[6]),
- the performance of the instrument,
- the observing procedures used to obtain the data,
- the scientific objectives of the original observing program.

The FLAMINGOS-2 instrument is very well documented on the Gemini/FLAMINGOS-2 website[7] and in published papers. But it is not necessary to understand every detail of the instrument design and operation to reduce your data; links to relevant portions of the instrument literature will appear throughout this manual.

[1] http://www.gemini.edu
[2] https://www.gemini.edu/sciops/instruments/gmos/?q=sciops/instruments/flamingos2
[3] http://www.astro.ufl.edu/
[4] https://www.gemini.edu/sciops/instruments/flamingos2/status-and-availability
[5] https://www.gemini.edu/sciops/data-and-results/gemini-observatory-archive
[6] https://www.gemini.edu/sciops/data-and-results?q=node/10797#Off-lineDP
[7] https://www.gemini.edu/sciops/instruments/flamingos2/

CHAPTER 1

Introduction

1.1 About This Cookbook

This cookbook provides python programs for reducing imaging and long-slit spectroscopic data taken with the FLAMINGOS-2 instrument on the Gemini South telescope, and descriptions of how to use and modify them. While these programs do group together reduction steps and can be run from start to finish to fully reduce scientific data with a minimal amount of preparation, they are *not* intended to be autonomous pipelines. You should always visually inspect your data for obvious defects and understand the processes involved in reducing your data so you can pinpoint where any problems arise.

No prior knowledge of python is assumed or required, although a basic understanding will be helpful: due to the rather awkward way in which PyRAF acts as an interface between python and IRAF, the tutorial scripts are not the best learning material.

1.2 Software Environment

The IRAF **gemini** package is presently the most comprehensive set of utilities for reducing data from F2. The only fully-supported way of obtaining the up-to-date version of this software is through the AstroConda[8] distribution channel of Anaconda.

Full details of how to obtain and install Gemini IRAF and its dependencies can be found on the Gemini website[9] and will not be addressed here.

If you already have AstroConda on your system, ensure that the packages are up to date:

[8] http://astroconda.readthedocs.io/en/latest/index.html
[9] http://www.gemini.edu/sciops/data-and-results/processing-software

```
conda update --all
```

1.2.1 Getting Help

Help is available if you run into problems. Contact:

- help@stsci.edu with problems installing or invoking the software
- Gemini Help Desk[10] with problems with using the software for data reduction.

1.3 Data Downloads

Search and retrieve the science and calibration data from the Gemini Observatory Archive[11] for each program, source, or calendar night of interest. See the GOA overview[12] for details.

1.3.1 First-Time Access

If you are a general archive user, no login is necessary to search for any data, or to retrieve non-proprietary data including calibration frames. **Only** if you are a PI or Co-I of an observing program **and** you wish to retrieve your *proprietary* data, you must do the following before you can access these files:

1. Request an account[13] if you don't already have one. You will receive an e-mail telling you how to establish a password (or *data access key* in their vernacular).
2. Navigate to the Gemini Observatory Archive[14] in your browser.
3. Click the `Not logged in` link at the upper right of the page and login using your account credentials. **You will need your program ID and password to retrieve your proprietary data.**

PIs of Gemini observing programs should have received in their award notification email the instructions for how to establish an account.

Caution: Note that applicable ancillary data (arc lamp, flat-field, or standard star exposures) may have been obtained on a night other than that of the science observation(s) of interest. You may need these exposures to calibrate the science data, particularly if they were obtained in Queue mode. These calibration data are normally accessible by clicking the **Load Associated Calibrations** tab following a successful search.

[10] http://www.gemini.edu/sciops/helpdesk/
[11] https://archive.gemini.edu/searchform
[12] https://www.gemini.edu/sciops/data-and-results/gemini-observatory-archive
[13] https://archive.gemini.edu/request_account/
[14] https://archive.gemini.edu/searchform

1.3.2 Archive Searches

Science Data

There will always be multiple ways to select the science data you want from the GOA search page, including via Program ID, Target Name, or celestial coordinates. Note that, although proprietary data will appear in a search, only *non-proprietary* archival data (or your own proprietary data) will be available for download (the proprietary period for Gemini data is currently 12 months).

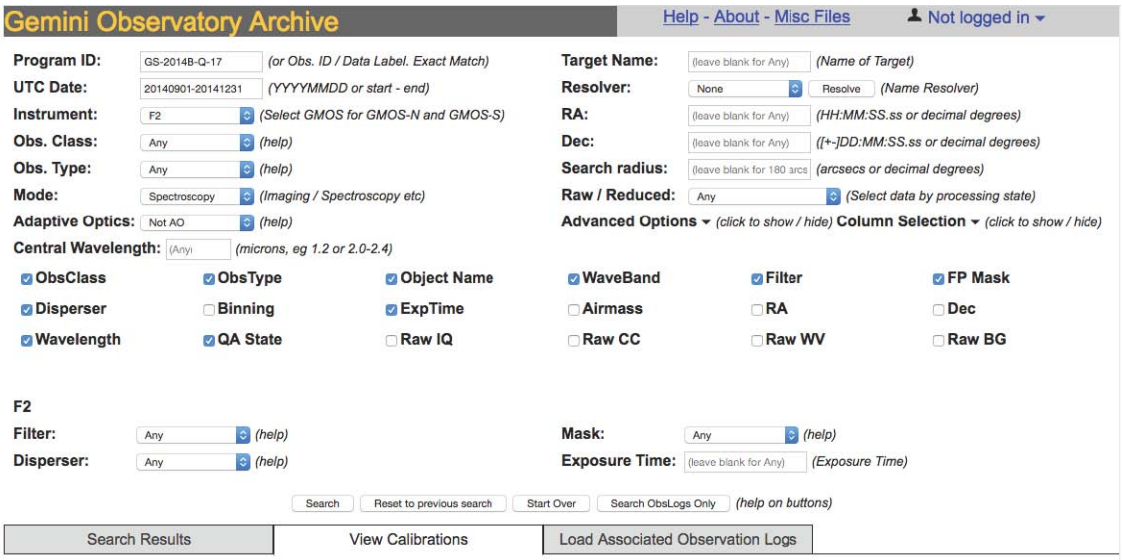

Fig. 1: Interface for GOA search for FLAMINGOS-2 data, also showing the available metadata that may be displayed in columns of the results table. The tabs at the bottom allow access to the calibration data for the specified program. This search was for spectroscopic data from program GS-2014B-Q-17 with a restricted date range.

After a successful search for your data of interest, you should scroll to the bottom of the search results and click the *Download all [NNN] files* button. This will create a tar of the selected files and download it to your local disk. If you are only interested in a few files, you can manually check those files and click the *Download Marked Files* button.

Calibration Data

Calibration exposures are routinely obtained by Gemini staff to support queue observations, and to monitor the health and performance of the instruments. The exposures of most potential interest for data reduction include:

- Darks
- Flat-fields
- Arcs
- Telluric standard stars

Very often observers include additional standard star exposures in their programs, depending upon the science goals.

To find the appropriate calibrations for your chosen science exposures, click the **Load Associated Calibrations** tab on the search results page. Note that this will find the calibrations appropriate for *all* the science exposures, and not just those that have been checked. It is therefore worth being as precise as possible in your science data search to ensure that only relevant calibrations are found.

Click the *Download all [NNN] files* button. It is common for these files to include some exposures you do not need, but but it is easier to ignore them during data reduction than to attempt to filter them out with tighter archive search criteria.

After downloading all files, you should create a working directory for the raw data, and extract the files from the tarballs there, using `tar xvf /path/to/gemini_data.tar`. Then use `bunzip2` to uncompress the files. If an unnecessarily large number of calibration files have been downloaded, this is a sensible time to delete any extraneous ones.

1.3.3 Types of Observations

The following types of FLAMINGOS-2 observations are routinely obtained, depending upon the observing program. Types in *italics* are rarely useful for data reduction.

Table 1: **Types of Observations**

Type	Frequency	Description
Dark	several per week	Sequence of finite-duration exposures with the shutter closed. Duration of darks must match the duration of science exposures and be taken with the same readout mode.
Flat-field	several monthly per filter	Sequence of exposures of the *GCAL* flat-field lamp. They are combined and normalized to apply the pixel-level sensitivity correction.
Comparison Arc	one or more per night per slit/grating combination	Exposures of the Argon comparison arc used to derive geometric rectification and wavelength calibration.
Image	one or more per filter per target field	**Science image** obtained with `ObsMode = imaging`. May also be obtained for target field acquisition. Usually these are dithered to allow background subtraction.
Acquisition image	one or more per target field	Short-duration image obtained through a custom *Slit-mask* (`ObsMode = acq`). Used to determine offsets from targets to slits; not used for data reductions.
Long-slit spectrum	one or more per target position	**Science spectrum** obtained with a facility longslit (`MASKNAME = <X>pix-slit`). Usually these are dithered along the slit to allow subtraction of bright sky lines.
MOS spectrum	one or more per target position	**Science spectra** obtained with a custom Slit-mask (`MASKNAME` = <mask>); one spectrum per slit including field stars. Mask names include the observing program ID.

1.4 Data Packaging

1.4.1 File Nomenclature

It is usually simplest during data reduction to retain the filenames of raw exposures as provided by the Gemini Observatory Archive, and to allow processing tasks to take care of naming output files. The raw filename template is the following:

<*site*><*yyyy*><*mm*><*dd*> `S` <*nnnn*> `.fits`

where `S` and `.fits` are literals, and:

- <*site*> is either `N` or `S`, indicating which telescope took the data
- <*yyyy*><*mm*><*dd*> is the year, numerical month, and UT date of observation
- <*nnnn*> is a 4-digit (prefixed with zeroes if necessary) running sequence number within a UT day

1.4.2 Multi-Extension FITS

FLAMINGOS-2 raw data, and processed data as produced by tasks in the **f2** and related packages, are stored in *FITS* files and structured internally in Multi-Extension FITS (*MEF*)—i.e., FITS files with one or more standard extensions[15]. MEF files are used to group logically connected data objects, as explained below and on the FLAMINGOS-2 website[16]. Each MEF file contains a Primary Header-data unit (*PHU*), followed by one or more standard FITS extensions[17]. The extensions are numbered sequentially, and will contain header keywords `EXTNAME` describing the type of data they contain and `EXTVER` with a value equal to the extension number.

F2 MEF files follow the *FITS* Standard recommendation that the PHU never contains image pixel data; the extensions are either of type IMAGE or BINTABLE, and no other type. The number and type of extensions in F2 data files depends upon the level of processing and the content, and the extensions can appear in any order. Raw exposures contain a $2048 \times 2048 \times 1$ pixel array in the first extension. The table below summarizes the structure of the contents for reduced data products. Optional extensions in grey are added if the `fl_vardq+` flags are specified during processing.

Extension	Type	Description
		All Files
[0]	PHU	File metadata as *keyword=value* records
		Raw Exposures
[MDF]	BINTABLE	Aperture definition table
[SCI,1]	IMAGE	Raw image array (2048x2048x1)
		Reduced Image
[SCI,1]	IMAGE	Science image array
[VAR,1]	IMAGE	Variance image array
[DQ,1]	IMAGE	Data quality image array (bit-encoded)
		Reduced Longslit Spectrum
[MDF,1]	BINTABLE	Aperture definition table
[SCI,1]	IMAGE	Science spectrum
[VAR,1]	IMAGE	Variance array
[DQ,1]	IMAGE	Data quality array (bit-encoded)
		Reduced MOS Spectrum
[MDF,1]	BINTABLE	Aperture definition table
[SCI,1]	IMAGE	Science spectrum for first slitlet
[VAR,1]	IMAGE	Variance array for first slitlet
[DQ,1]	IMAGE	Data quality array (bit-encoded) for first slitlet
		[additional SVD extension triplets for each slitlet]

[15] http://fits.gsfc.nasa.gov/xtension.html
[16] http://www.gemini.edu/sciops/instruments/flamingos2/status-and-availability/fixes-and-improvements-20092011
[17] http://fits.gsfc.nasa.gov/xtension.html

CHAPTER 2

Instrument Overview

To reduce data from FLAMINGOS-2 effectively, and to understand its limitations, it is helpful to know the basics of the design and operating modes. These instruments are very well documented on the Gemini Observatory website (see F2[18]) and in the literature (see *[EE3]*). This chapter contains a *very* brief summary of the instrument; links to more in-depth material appear throughout the discussion.

2.1 Optical Path

The optical path, illustrated in the schematic diagram below, shows relative positions of the MOS and dekker wheels, as well as wheels holding filters and grisms. The selection of aperture slit-mask, filter, and grism defines the observing configuration for each exposure.

2.2 Focal Plane

The FLAMINGOS-2 field of view (*FoV*) is circular with a diameter of 6.1 arcmin, which underfills the detector in imaging mode (see the Figure below). In spectroscopic MOS mode the field is truncated by the dekker to about 2 arcmin in the detector *x*- direction.

2.2.1 Detector

FLAMINGOS-2 uses a single Hawaii-2RG HgCdTe detector array, with the following characteristics:

[18] https://www.gemini.edu/sciops/instruments/flamingos2/

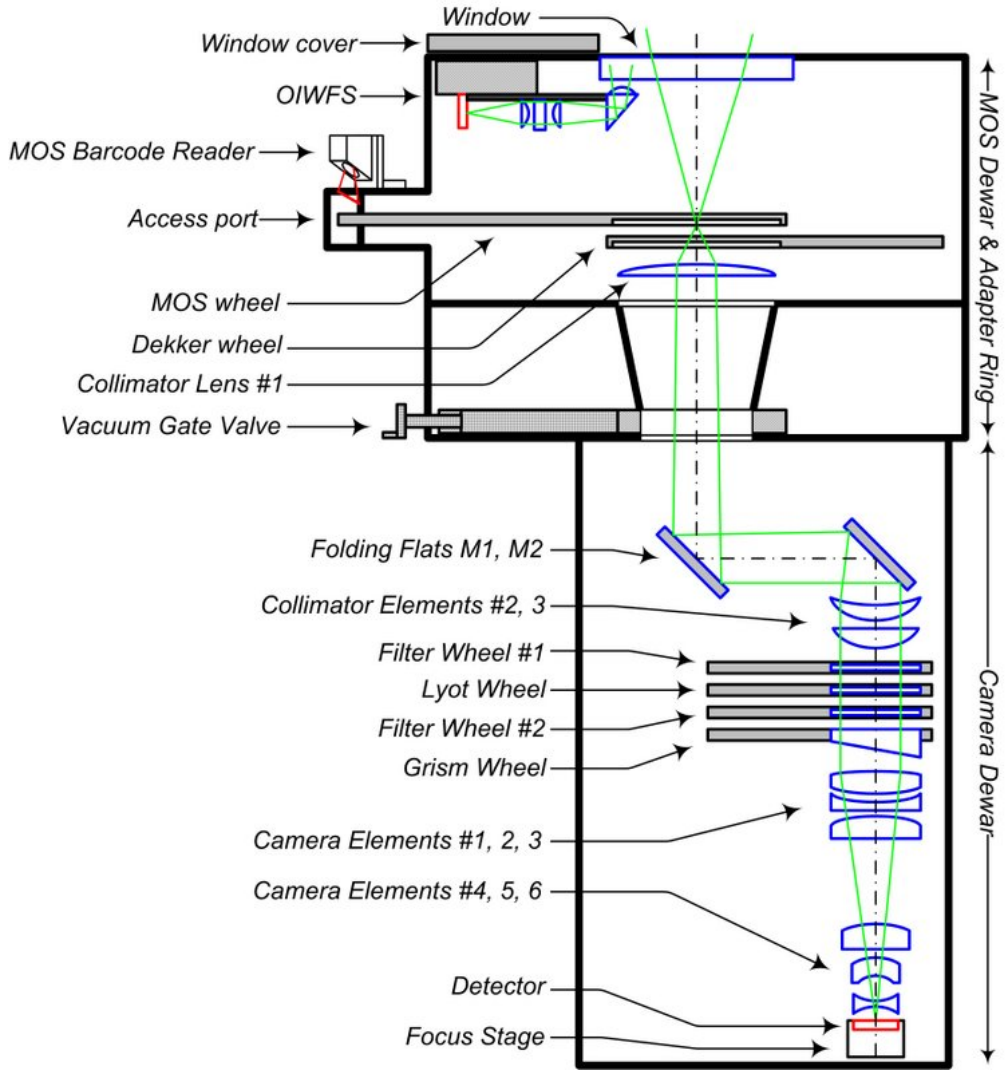

Fig. 1: FLAMINGOS-2 optical path. Light from the telescope focal plane enters from the top, and passes through a slit-mask or dekker. A filter and a grism may also be inserted into the beam, depending on the observing configuration, before light enters the camera and illuminates the detector (*bottom*). Click to enlarge.

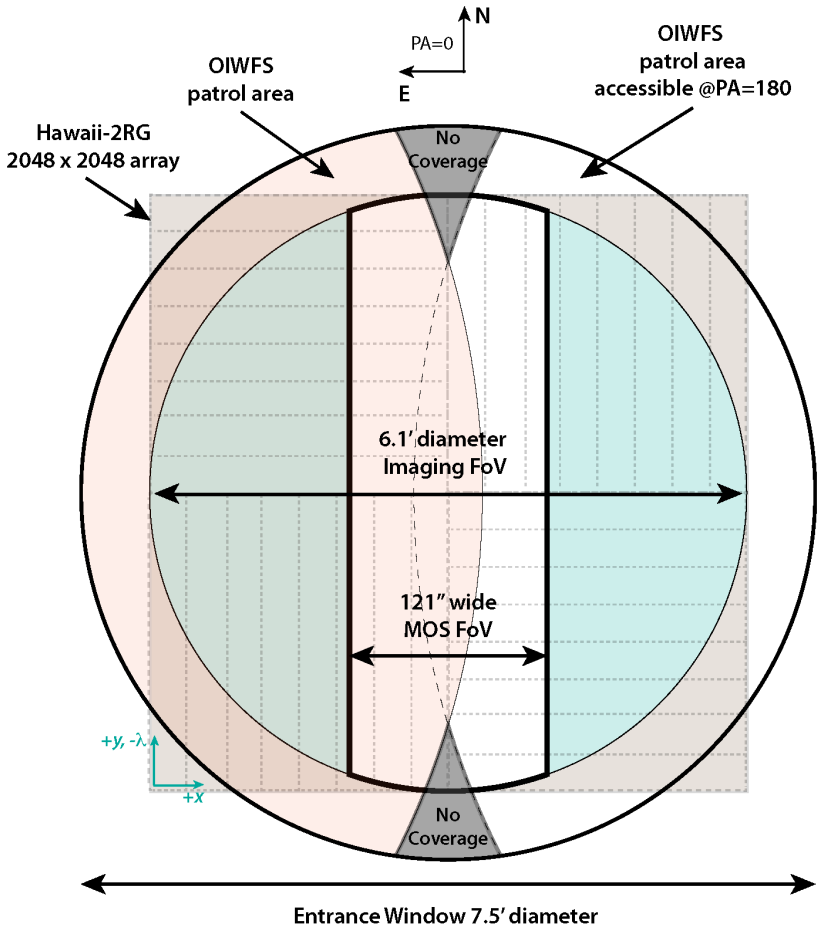

Fig. 2: The 6.1 arcmin circular Imaging FoV (*light blue*) and the 2 arcmin wide MOS (*white*) FoV lie within the spectrograph entrance window (*dark black circle*). The detector (*light brown*) is divided into 32 regions (*light dashed grey rectangles*) for parallel readout, with the raw image origin indicated at lower left (*teal arrows*). In imaging mode the corners of the detector are unilluminated. The extent of the OIWFS patrol area (*light orange*) is indicated for PA $= 0^\circ$ though a mirror-symmetric area is accessible at PA $= 180^\circ$. Two small areas inaccessible to the OIWFS are indicated (*dark grey*). Click to enlarge.

Table 1: **Hawaii-2RG Detector Characteristics**

Image format	$2048 \times$ 2048 array; $18\mu m$ pixels
Output channels	32
Spatial scale	0.1787 arcsec/pixel at f/16
Spectral response	$0.9 - 2.4\mu m$
Dark Current	0.5 $e^{-}\mathrm{s}^{-1}\mathrm{pix}^{-1}$
Read Noise	$11.7e^{-}$ (single CDS read)
	$< 5e^{-}$ (8 CDS read)
Gain	$4.44e^{-}\mathrm{ADU}^{-1}$
Well depth	$155400e^{-}$; 35000 ADU

The array response is linear to <0.5% from about 4000 - 22,000 ADU. See the F2 instrument pages for details.

The detector read-out may be optimized for the target source brightness: *bright-*, *medium-*, or *faint-object* mode. They differ in the number of correlated double-sample (CDS) reads of the array: 1, 4, and 8 respectively. All reads for an exposure are performed onboard the detector electronics, and only the final averaged pixel values are written to the raw image. The number of CDS reads affects the read-out time and the resulting read noise. At very high count rates the exposure duration per sample should be sufficiently short so that:

- the count rate remains in a regime where any non-linearity can be corrected
- minimal signal is lost during the time it takes to read the array

These conditions might be violated locally for, e.g., bright field stars.

2.3 Configurations

The FLAMINGOS-2 imaging spectrograph can be configured in the following ways:

- **Imaging** of a circular 6.1 arcmin *FoV*
- **Long-slit spectroscopy** with moderate resolution and a variety of slit widths
- Simultaneous **multi-object spectroscopy** of targets within a 6.1×2.0 arcmin portion of the imaging FoV, with custom fabricated slitlets

2.3.1 Imaging

A set of facility filters may be used to obtain IR images in a few passbands of the 6.1 arcmin circular FoV. Acquisition images are also obtained for all spectroscopic targets, but their short duration may limit their scientific utility. The facility filters are: *Y, J, H, Ks, K-blue, K-red*. Additional band-limiting filters, *Jlow*, *JH*, *HK*, and *K-long* are also available for spectroscopic modes.

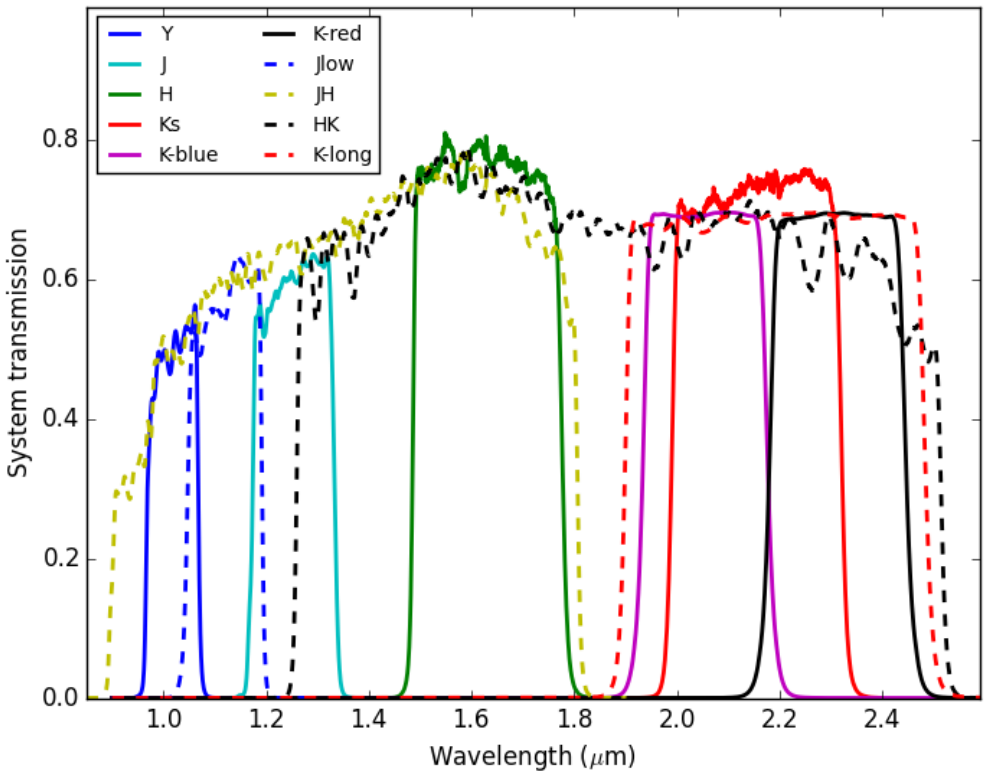

Fig. 3: Total system transmission of the facility filters (for K-blue, K-red, and K-long only the scaled filter transmission is plotted). Transmission of the band-limiting filters is shown with dashed curves. Quantitative filter descriptions are available on the Gemini F2 web pages.

2.3.2 Spectroscopy

Long-slit or multi-object spectroscopy requires obtaining an acquisition exposure with a the slit mask inserted, but without the disperser to ensure that light from all targets in the field passes through the intended slitlets. These *acq* exposures are not useful for data reduction. The grisms place the spectra from the slit(s) onto the detector format for subsequent exposures.

Aperture Masks

The F2 facility longslit masks each have a length of 263 arcsec, and have widths as shown in the table below. The spectral resolution degrades for widths larger than 2 pixels.

Table 2: **Longslit Widths**

Width (pix)	1	2	3	4	6	8
Width (arcsec)	0.18	0.36	0.54	0.72	1.08	1.44

There are also 9 slots available for MOS spectroscopy with custom masks. A schematic of the locations of the aperture masks (and the *open* position) in the MOS wheel is shown below.

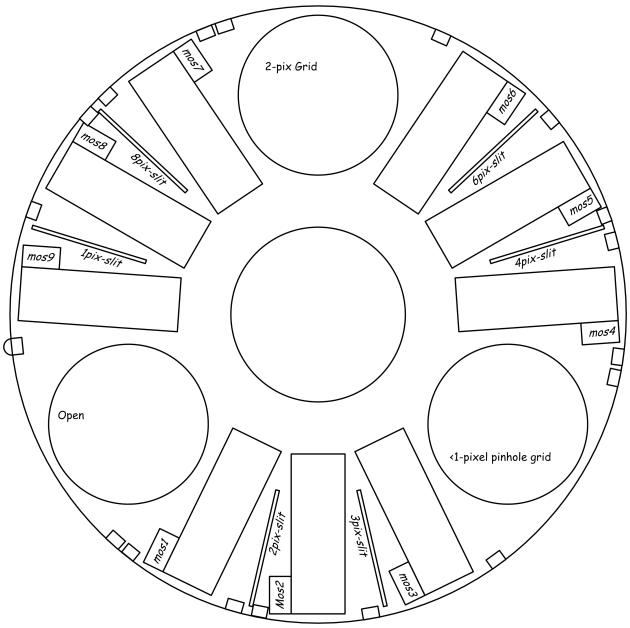

Fig. 4: Location of aperture masks in the F2 MOS wheel. Click to enlarge.

Blah.

Dispersers

Three grisms are available as dispersive elements in the spectroscopic configurations. Together with passband limiting filters, the grisms provide low- and intermediate-spectral resolution over nearly the full range

of the detector sensitivity. See the table below for their attributes, and the F2 website for more quantitative detail.

Table 3: **Available F2 grisms**

Grism	Filter	Order	Wavelength (μm)	Dispersion (A)	Resolution
JH	JH	1	1.25	6.677	900
HK	HK	1	1.65	7.826	900
R3K	Jlow	6	1.10	1.667	2700
	J	5	1.25	2.022	2800
	H	4	1.65	2.609	2800
	Ks	3	2.20	3.462	2900
	K-long	3	2.19	3.495	2900

The total system throughput of the commonly used grism + filter combinations is shown below.

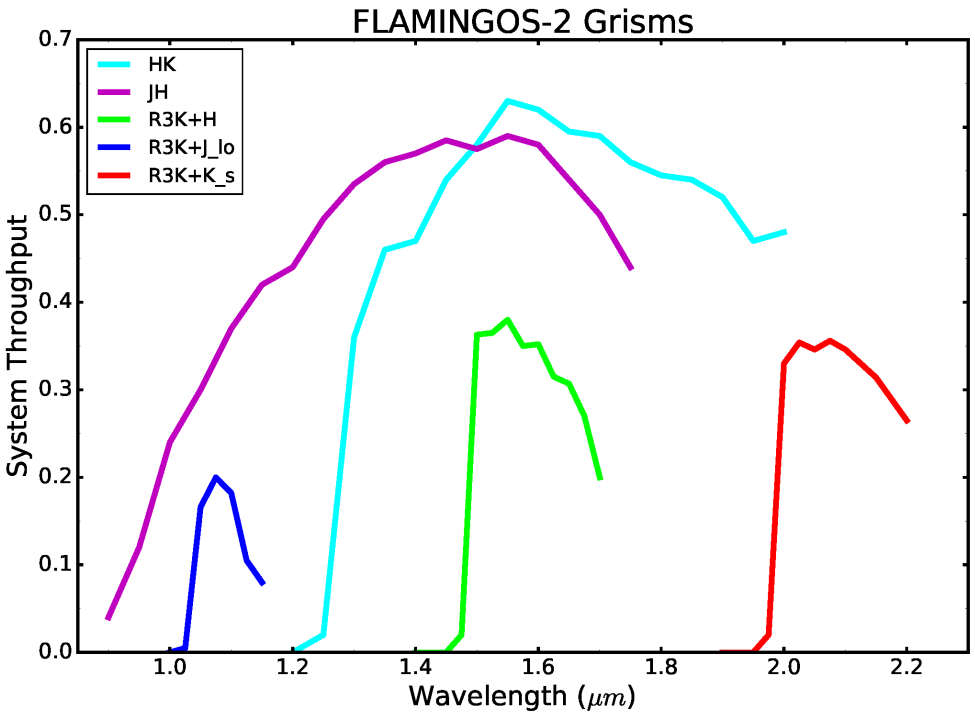

Fig. 5: Total system throughput of the F2 grisms. Throughputs include the transmission of the order-sorting filters.

CHAPTER 3

Data Processing Basics

3.1 Calibrations

Master calibration reference (*MasterCal*) files are derived from calibration or science observations, and are used to remove the various components of the instrument signature from the data. Calibration exposures may combined or characterized to create a calibration, so that it may be applied when science data are processed. Some other instrument calibrations (e.g., slit mask definition files for MOS mode) have already been created for you by Gemini scientists, or are distributed with the **gemini.f2** package.

3.1.1 Darks

Dark images are required for both imaging and spectroscopic observations, and show the spatially variable signal that accumulates during exposures in the absence of external illumination. This additive signal originates from multiple sources, including thermal radiation from both the shutter and the read-out electronics. The structure depends upon the exposure time and the read-out mode — i.e., the number of correlated double-sampled reads — for `bright` (1), `medium` (4), or `faint` (8).

The dark correction is applied by simply subtracting the matching **Dark MasterCal** using either **gemarith**, or one of **nireduce** or **nsreduce** depending upon the observing configuration.

It is best to co-add several (10 or more) dark exposures, obtained on the same night, so that the noise in the **Dark MasterCal** does not dominate in science exposures with low background. The convention in the tutorials is to name the **Dark MasterCal** files `MCdark_NNN` where `NNN` is the exposure duration in seconds. The output **Dark MasterCal** file will have one FITS image extension, or 3 extensions if you elected to create the VAR and DQ arrays. Darks are the same for both imaging and spectroscopic observations.

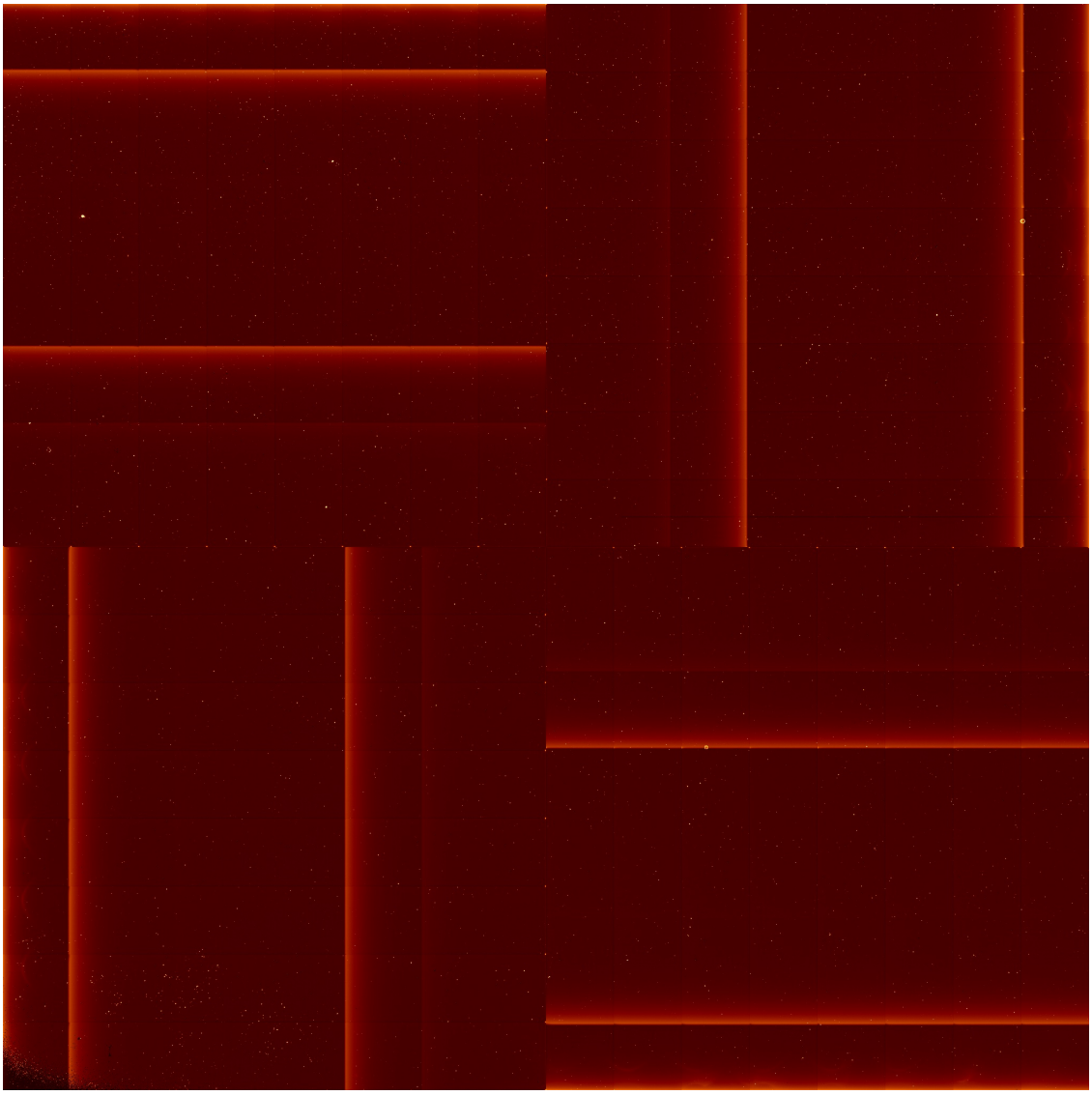

Fig. 1: **Dark MasterCal** in false-color with log intensity stretch for exposures of 60s duration and `READMODE = bright`. Note the amplifier glow along the edges of the 32 sub-arrays of the detector. Click image to enlarge.

3.1.2 Flat-Fields

Flatfields (required for both imaging and spectroscopy) are used to correct for differences in the sensitivities of pixels, to ensure that the same amount of illumination produces the same signal at all locations. Constructing a **Flat-field MasterCal** is largely a matter of combining dark-corrected flat-field exposures, with appropriate scaling, and outlier rejection. Flats for spectroscopy are obtained from observations of the *GCAL* continuum lamp, while imaging flats may be constructed from either lamp or night-sky observations. Separate flats must be created for each filter (imaging) or grism (spectroscopy); for *MOS* observations, separate flats are also required for each slit mask.

As with darks, it is best to combine a few to several well exposed flat-field exposures (if available) to keep noise in the flat-field from dominating the uncertainties in well exposed portions of the science data. The convention in the tutorials is to name the **Flat MasterCal** files `MCflat_NNN` where `NNN` is the name of the filter (imaging) or dispersing grism (spectroscopy).

Imaging Flats

GCAL imaging flats are usually created by subtracting exposures with the continuum lamp off from exposures with the lamp on. For *K* and *Ks*-band observations, however, the thermal emission is high and flats are made by subtracting dark exposures from exposures with the lamp off.

Flats can also be created from dark-subtracted sky exposures but, in these cases, it is necessary to mask objects before combining.

Long-Slit Flats

Spectroscopic flatfields are created from dark-subtracted spectra of the GCAL continuum lamp. Rather than simply divide the science exposures by these flats, the variation in sensitivity with wavelength must first be removed. This is achieved by fitting a smooth function along the wavelength direction and dividing through by this function. See *Flatfields* for more details.

3.1.3 Wavelength Calibration

Spectroscopic observations *only* need to be wavelength calibrated. Exposures of an Argon arc lamp are used to determine the dispersion solution for spectroscopic modes. Arc lamp exposures should always be dark corrected and, while not essential, a flat-field correction typically improves the fit at the ends of the spectrum.

Individual arc lines are automatically identified based on their groupings and the estimated wavelength solution, and an analytic function (typically a 4th or 5th order polynomial) is fit for the wavelength as a function of pixel location. Once completed, the individual lines are then traced in the spatial direction to determine additional non-linearities. This information is attached to the science exposures with the **gnirs.nsfitcoords** task, and the transformations are performed by **gnirs.nstransform**.

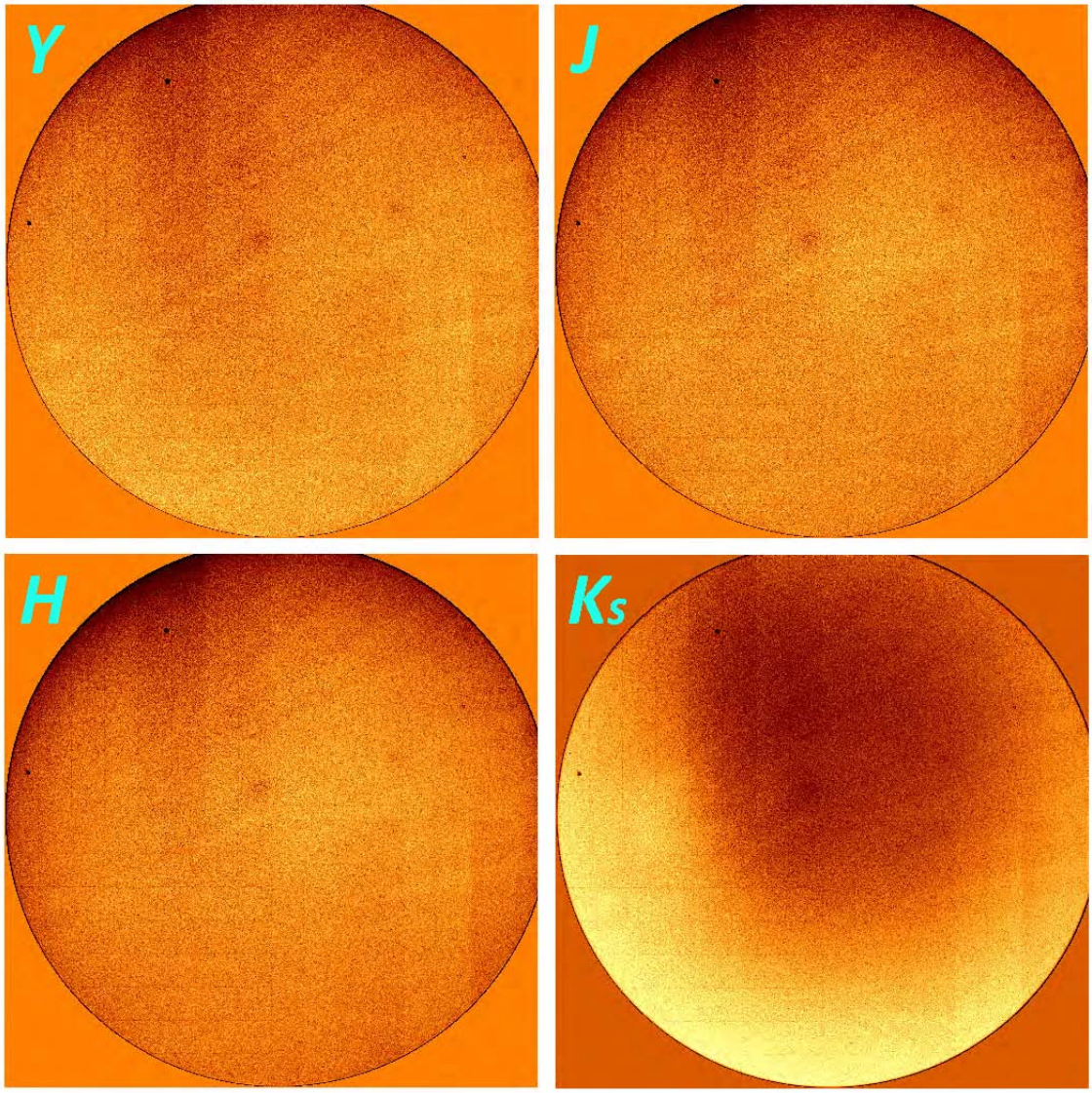

Fig. 2: Imaging **Flat-field MasterCals** for the *Y, J, H, Ks* filters. This false-color rendering has a linear intensity stretch and a range of $\pm$30 % (40% for Ks) about a mean of 1.0. Click image to enlarge.

3.1.4 Telluric Correction

Spectra of targets may need to be corrected for absorption by the Earth's atmosphere (*telluric absorption*). This can be derived from telluric standards (stars that have few, relatively weak features in the IR), provided they are obtained at similar airmass close in time. In a similar matter to the spectroscopic flatfields, a smooth function is fit to the reduced, extracted spectrum of the standard (ignoring strongly absorbed regions) to produce the fractional absorption as a function of wavelength, and this is then applied to the science spectrum. This is performed by the task **gnirs.nstelluric**.

3.1.5 Night-sky frames

The night-sky background in the infrared is composed of emission lines and continuum, with the latter increasing strongly with wavelength. This background very often dominates the brightness of astrophysical targets, and it is variable on timescales of minutes. The thermal background from the telescope and instrument is also fairly strong in the K-band, but it is fairly stable.

The sky illumination in near-infrared images is determined by making sky frames. Multiple sky images at different locations are median-combined with objects masked out to produce an image devoid of astronomical sources. For crowded fields or very extended targets, these may require *offset sky* pointings; in sparse fields, a *dither pattern* is often used with small steps that move the target(s) around on the detector.

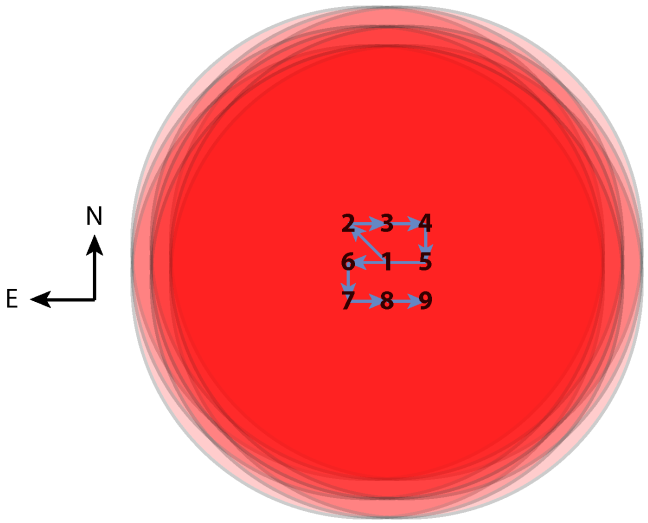

Fig. 3: Nine-position dither pattern for IR imaging, for which offset exposures of equal duration are obtained sequentially in the serpentine order shown. Depth of combined exposure of the target field is represented by the intensity of the red colored area.

3.1.6 Flux Calibration

Flux calibration for imaging observations is performed using fully-reduced images of *photometric standard stars*, whose brightnesses have been accurately measured. Gemini keeps a List of Photometric Standards[19] that are observed regularly to provide absolute photometric calibration of near-infrared images.

[19] https://www.gemini.edu/sciops/instruments/nearir-resources/photometric-standards

Spectrophotometric flux calibration is performed by dividing the fully-reduced spectrum of a star by a model representing its true spectrum in absolute flux density. This gives the factor for converting counts to flux density at each wavelength pixel, which can then be applied to the science spectrum.

At near-infrared wavelengths, there is little variation between the spectra of stars of a given spectral type, so almost any star can be used to perform this step. In practice, it is therefore possible to use the telluric standard as a spectrophotometric standard and combine the telluric correction and flux calibration into a single step.

3.2 Using the Python scripts

The python code on which the tutorials are based has dependencies on some common **python** packages, as listed in the table below:

Table 1: **Python Package Dependencies**

File	Description
numpy[20]	Numerical operations on arrays
astropy[21]	General astronomical utilities including FITS I/O
yaml[22]	Data serialization language for configuration files

These packages are included by default in the Anaconda distribution of python[23], which is highly recommended.

The reduction scripts are written as self-contained python programs, which can be executed in *either* of the following two ways:

```
python f2_reduce_images.py
./reduce_images.py
```

(the second will only work if the program has executable permission). A sensible way to run the reduction is one step at a time, commenting out the other steps, and examining the output before proceeding. For a more interactive session, you can choose to start up PyRAF (or python) and import the functions, which you will then be able to call directly:

```
% pyraf
--> from reduce_images import *
```

Each step of the data reduction is written as two python functions, typically appearing as

```
flat_dict = selectFlats(obslog)
reduceFlats(flat_dict)
```

The first of these queries the *Observing Log* to determine which unique flatfields can be made from the data, and which raw input frames and pre-existing calibrations (e.g., dark frames) are needed to make them. This

[20] http://www.numpy.org
[21] http://www.astropy.org
[22] https://martin-thoma.com/configuration-files-in-python/#yaml
[23] https://store.continuum.io/cshop/anaconda/

is returned as a python dictionary in a format that can be passed to a function that performs the actual reduction by stringing together various IRAF tasks. This is somewhat inefficient from a coding perspective but it makes it much simpler to adapt the code by, for example, creating the dictionaries directly. Understanding python dictionaries will help you to get the most out of the scripts.

3.2.1 Python dictionaries

A python dictionary is an *unordered* list of pairs of *keys* and *values*. The reduction dictionary at each stage of processing contains entries where the key is the name of the output file, and the value is itself a dictionary describing the input files, with keys indicating the type of file and values giving the filenames. (For flatfield reduction, the bad pixel mask is included in the dictionary, even though it is an *output* file.)

If you are only reducing a small number of files, you may wish to make your dictionaries by hand to ensure maximum control over them. You initialize a dictionary as follows:

```
dict = {key1: value1, key2: value2, ...}
```

and add to or update it in either of the following ways:

```
dict[key3] = value3
dict.update({key3: value3, key4: value4, ...})
```

Let's suppose you are reducing spectroscopic observations and you have the following data:

Table 2: **Sample data**

Filenames	Type of data
S20180101S0001-S20180101S010	Darks
S20180101S0011-S20180101S020	GCAL flats
S20180101S0021	Arc

You can make the required python dictionaries as follows:

```
raw_darks = ['S20180101S{:04d}'.format(i) for i in range(1, 11)]
dark_dict = {'MCdark': {'input': raw_darks}}
raw_flats = ['S20180101S{:04d}'.format(i) for i in range(11, 21)]
flat_dict = {'MCflat': {'dark': 'MCdark',
                        'bpm': 'MCbpm.pl',
                        'input': raw_flats}}
arc_dict = {'MCarc': {'dark': 'MCdark',
                      'bpm': 'MCbpm',
                      'flat': 'MCflat',
                      'input': ['S20180101S0021']}}
```

Note the input for the arc is a list, even though only a single file is being used. Also note how a list of consecutive filenames is constructed for the darks and flats, and how the second number is one more than the last frame to be included.

One final piece of good-to-know python information is that a statement is assumed to carry onto the next line of the file if any parentheses, brackets, or braces have not been closed.

Task parameter files

PyRAF stores the parameters for the individual IRAF tasks as dictionaries. Each reduction step resets these parameters to their IRAF defaults with the `unlearn()` function and then reads in new values from a dictionary stored on disk as a `yaml` file. For example, the `imgTaskPars.yml` file starts like this:

```
f2prepare:
    rawpath: ./raw
    outprefix: p
    logfile: f2prepLog.txt

gemarith:
    fl_vardq: 'yes'
    logfile: gemarithLog.txt
```

The name of the task is given, followed by only the parameters that differ from the defaults (or whose values are most likely to have an effect on the final data products). Here, we are telling **f2prepare** that the raw files live in a subdirectory, and we wish to use the prefix `p` (rather than `f`) for prepared files, and log the task's actions to a specific file.

The `get_pars()` function provided in the scripts performs several steps. First, it "unlearns" the specified tasks to set the parameter values back to their IRAF defaults. It then constructs dictionaries of overrides from the `yaml` file before returning them.

3.2.2 File naming conventions

The Gemini convention for naming output files is to prepend one or more characters to the input filename. This occurs for each intermediate stage of data reduction processing, and is summarized in the table below. Unfortunately the characters used are not entirely unique, so the meaning of a few of them must be derived from context.

Table 3: **Processing Prefixes**

Prefix	Applies to:	Description
a	Spec	telluric correction applied
c	Spec	Flux calibrated
d	Img+Spec	Dark-subtracted
f	Img+Spec	Flatfielded
p	Img+Spec	Prepared
r	Img+Spec	Arbitrary reduction with nireduce/nsreduce
t	Spec	transformed (wavelength-calibrated rectilinear spectral image)
x	Spec	extracted 1-D spectra

During the processing, there will typically be a step where multiple input files are combined into a single output file. In such cases, the output file is normally given a completely new name. Where the output is a master calibration file, the style in the Tutorials is to use the format

`MC` <*caltype*> `_` <*config*> `.fits`

where:

- <*caltype*> is the type of calibration, e.g., `dark` or `flat`
- <*config*> provides a unique configuration, the details of which will depend on the type of calibration. For a dark, this may simply be the exposure time, while for a flatfield it will be the filter. Additional information can be added, such as the date if separate calibrations are needed for each observing night.

For on-sky exposures of science targets or standard stars, a unique name should be given that will depend on the observing strategy.

3.3 Observing Log

How you organize your data is up to you and may depend on the number of files you have. The tutorials in this cookbook assume that you have created a working directory for the reduced files, with a single `raw/` subdirectory into which all the raw files have been placed. In order to keep track of the files, it will be necessary to create an observing log that holds the important metadata for each file. The tutorials make use of this log to determine which files should be used at each stage of the reduction process in an automated manner.

3.3.1 Creating the Observing Log

A python script is provided that creates an observing log and writes it to disk as a FITS table. To use it:

- Download `obslog.py`
- Navigate to the `raw/` subdirectory containing the raw files
- Type `python /path/to/obslog.py obslog.fits`

The script opens the raw files in the directory in sequence and extracts relevant metadata from the primary header. The log can be viewed and edited with any software capable of handling FITS tables, such as TOPCAT[24]. In addition to the columns containing the file metadata, there is a column titled `use_me`. This can be unchecked to remove files from consideration by the automated reduction steps in the tutorials.

3.3.2 Header Metadata

Values from the keywords listed below are harvested from the FITS headers. Some of the names are obscure, so they are re-mapped to somewhat more intuitive field names in the Observing Log. Fields may be added (or deleted: *not recommended*) by changing the KW_MAP definition at the top of the python script.

[24] http://www.star.bris.ac.uk/~mbt/topcat/

Table 4: **Header Metadata**

Field name	Keyword	Description
use_me		Flag indicates file usage or exclusion (`True|False`)
File		Filename (excluding `.fits`)
Object	`OBJECT`	Name of target
Filter	`FILTER`	Name of filter (imaging or blocking)
Disperser	`GRISM`	Name of dispersing element
ObsID	`OBSID`	Observation ID (e.g. GS-2018A-Q-1)
Texp	`EXPTIME`	Exposure time (in seconds)
Date	`DATE-OBS`	UT Date of observation start (YYYY-MM-DD)
Time	`TIME-OBS`	UT Time of observation start (HH:MM:SS.S)
RA	`RA`	Right Ascension of target (deg)
Dec	`DEC`	Declination of target (deg)
RA Offset	`RAOFFSET`	Offset in Right Ascension from target (arcsec)
Dec Offset	`DECOFFSE`	Offset in Declination from target (arcsec)
ObsType	`OBSTYPE`	Type of observation: (`arc|cal|dark|flat|mos|mask|pinhole|ronchi|o`
ObsClass	`OBSCLASS`	Class of observation: (`acq|acqCal|dayCal|partnerCal|progCal|scienc`
Read Mode	`READMODE`	Detector readout Mode (`Bright|Medium|Dark`)
Reads	`LNRS`	Number of non-destructive reads
Coadds	`COADDS`	Number of array coadds
Mask	`MASKNAME`	Name for selected slit(mask)
MaskType	`MASKTYPE`	Type of mask (0=slit; 1=MOS; -1=pinholes)
Decker	`DECKER`	Decker position
GCAL Shutter	`GCALSHUT`	Position of GCAL shutter (`OPEN|CLOSED`)
PA	`PA`	Position angle of instrument
Wavelength	`GRWLEN`	Grating approximate central wavelength (:math:‘mu‘m)
Airmass	`AIRMASS`	Airmass at time of observation

3.3.3 Using the Observing Log

The observing log can be queried in a simple manner by selecting observations whose metadata match supplied values. Specific examples are shown in the tutorials, but a brief reference is presented here. Some basic familiary with python is required.

The `ObsLog` class is defined at the start of each tutorial file, and is loaded with the syntax

```
obslog = ObsLog('path/to/obslog.fits')
```

To extract the metadata for a given file, use the syntax:

```
metadata = obslog['S20180101S0001']
```

Specific items of metadata can be extracted with:

```
t = obslog['S20180101S0001']['Texp']
t, obstype = obslog['S20180101S0001']['Texp', 'ObsType']
```

To find observations that match a specific set of metadata, construct a python dictionary indicating the required matches, e.g.,

```
qd = {'ObsType': 'DARK', 'Texp': 5}
matching = obslog.query(qd)
```

will return all rows containing 5-second dark exposures. To return only the filenames of these exposures, use:

```
darkfiles = obslog.file_query(qd)
```

To select observations from a particular UT date, or between two (inclusive) dates, use:

```
matching = obslog.query({'Date': '2018-01-01'})
matching = obslog.query({'Date': '2018-01-01:2018-01-03'})
```

Finally, the `first` and `last` keywords can be used to select the first and last filenames, e.g.,

```
qd = {'ObsType': 'FLAT', 'Texp': 10, 'first': 'S20180101S0020',
      'last': 'S201801020355'}
files = obslog.file_query(qd)
```

will return the filenames of all 10-second flatfield exposures in the range specified.

An additional python function, `merge_dicts()`, is provided to assist with the construction of queries. It takes two dictionaries as arguments and returns a single dictionary by using the second dictionary to add new entries (only if `allow_new=True`) or update existing entries in the first dictionary.

Advanced Usage

The `ObsLog` class simply allows a very limited number of simple methods on a data table. Queries return another table, which can be used to instantiate a new observation log, e.g.,

```
obslog_night1 = ObsLog(obslog.query({'Date': '2018-01-01'}))
```

For queries that are more complicated than simple matching, the table itself is accessible as the `table` attribute of the log. For example, suppose you want to select all science exposures with offset distances greater than 60 arcseconds:

```
# Extract offset information
raoff, decoff = obslog.table['RA Offset'], obslog.table['Dec Offset']
distance_squared = raoff*raoff + decoff*decoff
# Make a new log of objects more than 60 arcseconds away
newlog = ObsLog(obslog.table[distance_squared > 3600])
# Now query this log
distant_files = newlog.file_query({'ObsClass': 'science'}))
```

CHAPTER 4

F2 Imaging Tutorial

The recipe described here provides a recommended, **but not unique** path for processing your FLAMINGOS-2 science data.

This tutorial will use observations from program GS-2013B-Q-15 (PI:Leggett), NIR photometry of the faint T-dwarf star WISE J041358.14-475039.3, obtained on 2013-Nov-21. Images of this sparse field were obtained in the *Y,J,H,Ks* bands using a dither sequence; *dayCal* darks and *GCAL* flats were obtained as well. Leggett, et al. (2015; *[L15]*) briefly describe the data reduction procedures they followed, which are similar to those described below.

4.1 Retrieving the Data

The first step is to retrieve the data from the Gemini Observatory Archive (see *Archive Searches*). The observations of this target were taken with a single Observation ID, so it is possible to obtain all the science exposures with a single query:

```
https://archive.gemini.edu/searchform/GS-2013B-Q-15-39
```

After retrieving the science data, click the **Load Associated Calibrations** tab on the search results page and download the associated dark and flat-field exposures.

If you look carefully, you will note that each set of flatfields comprises 6 frames from 2013-Nov-26 and 4 from 2013-Nov-29. F2 flats are typically done in sets of 12 (6 each of lamp-on and lamp-off, except for the *K* band), so the fact that the archive search has had to use observations from two different nights indicates a problem. Better is to use the flatfields from the later night, and these can be retrieved with the following URL:

```
https://archive.gemini.edu/searchform/F2/imaging/20131129/FLAT
```

The flats from 2013-Nov-26 (the only files from this date) can be deleted from disk in the normal way:

```
rm S20131126*
```

You will also note that there some pre-reduced calibrations exist in the archive. These are identified with the original name of the first file and a suffix indicating the type of calibration, e.g., `S20140124S0093_dark.fits`. These can be left as the observing log will only include raw files.

4.1.1 Exposure Summary

The data contain exposures of a specific science target and *dayCal* calibrations; see the table below for a summary. The science exposures were obtained in a 3×3 spatial dither pattern, with a spacing of about 15 arcsec in each direction from the initial alignment (see *Night-sky frames*).

Table 1: **Exposure Summmary**

Target	Filter	T_exp	N_exp
WISE 0413-4750	Y	120	9
	J	60	9
	H	15	72
	Ks	15	72
Dark		120	10
		60	21
		20	20
		15	10
		8	24
		3	13
GCAL Flat	Y	20	12 (6 on + 6 off)
	J	60	12 (6 on + 6 off)
	H	3	12 (6 on + 6 off)
	Ks	8	6 (off)

Note that dark exposures have been taken with exposure times that match not just the science exposures, but also the flatfields. Darks for F2 are usually taken in groups of 10, and you may choose to use only the 10 darks taken closest in time to the relevant exposure; if so, you can either delete the additional exposures now or uncheck the `use_me` flag after creating the observing log.

4.2 Preparation

First download `obslog.py` to the `raw` subdirectory and create an observing log, as described in *Observing Log*.

```
python obslog.py obslog.fits
```

The other files needed for this tutorial are a python script and two configuration files.

- Download: `reduce_images.py`

This python script will perform an automated reduction of the WISE 0413-4750 data; see the section *Using the Python scripts* to understand how to use it. This tutorial will take you through it, step by step, so you can understand the procedure and how to edit it for your own F2 imaging data, should you choose to do so.

Configuration files are required for the IRAF task parameters that differ from the defaults, and to provide the script with information about the targets.

- Download IRAF task parameters: `imgTaskPars.yml`
- Download target information: `imgTargets.yml`

4.2.1 Target configuration file

Each entry in this file gives the name (or root name) of the output file, and is followed by a list of parameters that will query the observation log to produce the list of science input frames. In addition, a parameter `groupsize` can be added, which will break the list of science frames into groups of this size, each of which is reduced independently (see *Science targets* for more details). Note that, because only one object has been observed in this program, only the filter needs to be specified in the configuration file. Since all exposures were taken on the same night, we use the default global darks, flats, and BPMs, and do not need to specify them in the file.

```
Y0413:
    Filter: Y

J0413:
    Filter: J

H0413:
    Filter: H
    groupsize: 9

K0413:
    Filter: Ks
    groupsize: 9
```

4.2.2 Configuration of nireduce

The **nireduce** task has several parameters; the table below lists the defaults for the processing flags — i.e., the parameters with logical values to indicate whether to perform an operation.

Table 2: **nireduce Processing Flag Defaults**

Flag	Default	Description
`fl_autosky`	No	Determine constant sky level to restore?
`fl_dark`	Yes	Subtract dark image?
`fl_flat`	No	Apply flat-field correction?
`fl_scalesky`	Yes	Scale the sky image to input image?
`fl_sky`	No	Perform sky subtraction using skyimage?
`fl_vardq`	Yes	Propagate VAR and DQ extensions?

The parameter values need to be chosen carefully, as the order of operations performed by the task is not consistent with the order adopted in this tutorial. For example, **nireduce** performs sky subtraction *before* flatfielding when both are selected, requiring the sky frame to *not* have been flatfielded, but this is not ideal for two reasons: it is difficult to determine how well objects have been removed from the sky frame without it having been flatfielded, and scaling is less reliable when the background counts are not uniform across the image. Therefore **nireduce** will be invoked multiple times, with different processing flag settings, to accomplish the processing steps in the needed order. Also, the simple process of subtracting a dark frame will be performed with the **gemarith** task, rather than **nireduce**.

4.3 Darks

Dark MasterCals are produced by combining individual dark frames. The function `selectDarks()` automatically produces lists of all dark frames of the same exposure time by querying the observing log.

```
def selectDarks(obslog):
    dark_dict = {}
    qd = {'ObsType': 'DARK'}
    exptimes = set(obslog.query(qd)['Texp'])
    for t in exptimes:
        darkFiles = obslog.file_query(merge_dicts(qd, {'Texp': t}))
        outfile = 'MCdark_'+str(int(t))
        dark_dict[outfile] = {'input': darkFiles}
    return dark_dict
```

This works by first querying the observing log for all dark frames. With the query dictionary `qd = {'ObsType': 'DARK'}`, `obslog.query(qd)` returns all rows in the observing log corresponding to dark frames. `obslog.query(qd)['Texp']` returns the exposure times of these frames, and the `set()` function collapses this down to a list of *unique values*.

Then there is a loop over each unique exposure time, with the observing log being queried for the names of files that are darks *and* have the correct exposure time. An entry is placed in `dark_dict` with an appropriately-named output file and the list of all raw dark frames. The exposure time is coerced to an integer because PyRAF has issues if there is a `.` in the name of a file.

There is also a function, `nightlyDarks()`, that will separate the darks by observation date as well as exposure time, which you might wish to do for your data (e.g., if you have observations, and darks, widely separated in time). This produces filenames like `MCdark_20180101_5.fits`, and you will have to adapt the subsequent code to select the correct one.

```
def reduceDarks(dark_dict):
    prepPars, combPars = get_pars('f2prepare', 'gemcombine')
    for outfile, file_dict in dark_dict.items():
        darkFiles = file_dict['input']
        for f in darkFiles:
            f2.f2prepare(f, **prepPars)
        if len(darkFiles) > 1:
            gemtools.gemcombine(filelist('p', darkFiles), outfile,
                                **darkCombPars)
        else:
```

(continues on next page)

(continued from previous page)

```
            iraf.imrename('p'+darkFiles[0], outfile)
    iraf.imdelete('pS*.fits')
```

Reduction of the dark frames is straightforward: they are prepared and then combined. If only one dark frame is sent, then it is simply renamed after being prepared.

4.4 Flatfields

Flatfield frames can be constructed either from observations of the calibration (GCAL) lamp, or from images of the sky (with the removal of astronomical objects).

4.4.1 GCAL Flats

As discussed in *Flat-Fields*, GCAL flats normally consist of observations of equal time taken with the shutter open ("lamp-on"), and with the shutter closed ("lamp-off"). For the *K* and *Ks* filters, only closed-shutter flats are taken and dark frames are subtracted from these.

```
def selectGcalFlats(obslog):
    qd = {'ObsType': 'DARK'}
    tshort = min(obslog.query(qd)['Texp'])
    shortDarks = obslog.file_query(merge_dicts(qd, {'Texp': tshort}))

    flat_dict = {}
    qd = {'ObsType': 'FLAT'}
    params = ('Filter', 'Texp')  # Can add 'Date'
    flatConfigs = unique(obslog.query(qd)[params])
    for config in flatConfigs:
        filt, t = config
        config_dict = dict(zip(params, config))
        if filt.startswith('K'):
            lampsOn = obslog.file_query(merge_dicts(qd, config_dict))
            lampsOff = obslog.file_query({'ObsType': 'DARK', 'Texp': t})
        else:
            config_dict['GCAL Shutter'] = 'OPEN'
            lampsOn = obslog.file_query(merge_dicts(qd, config_dict))
            config_dict['GCAL Shutter'] = 'CLOSED'
            lampsOff = obslog.file_query(merge_dicts(qd, config_dict))
        bpmFile = 'MCbpm_'+filt+'.pl'
        outfile = 'MCflat_'+filt
        flat_dict[outfile] = {'bpm': 'MCbpm_'+filt+'.pl',
                              'lampsOn': lampsOn, 'lampsOff': lampsOff,
                              'shortDarks': shortDarks}
    return flat_dict
```

The **niflat** task produces a bad pixel mask as well as the flatfield. In order to do this, it needs short-exposure darks, so the observation log is first queried for all the exposure times of all the dark frames; the lowest of these is determined and a second query made to find the list of dark files matching this exposure time.

The observation log is queried for flatfield images and the configurations of these (here the combination of filter and exposure time) are whittled down to a list of unique pairs – note that the `unique()` function must be used instead of `set()` when more than one field is being extracted from the log. For each configuation, the observation log is then queried two more times, to separate the lamp-on and lamp-off flats. For most filters, this is done by searching for flats with the appropriate combination of filter and exposure time, and the GCAL shutter either open or closed; for *K* and *Ks*, *any* flats are selected as lamp-on, while dark exposures of the same exposure time are used for the lamp-off exposures.

An entry in the reduction dictionary is then created, keyed by the name of the output file. Its value is a dictionary with the name of the *output* BPM file, and lists of the lamp-on and lamp-off files, and a list of the short-exposure darks.

4.4.2 Sky flats

Flatfields can also be made from the twilight sky. The same reduction dictionary format is used, but the sky images take the place of the lamp-on frames, and darks of the same exposure time are used in place of the lamp-off frames. The short darks are identified and used in exactly the same way as above.

```
def selectSkyFlats(obslog):
    qd = {'ObsType': 'DARK'}
    tshort = min(obslog.query(qd)['Texp'])
    shortDarks = obslog.file_query(merge_dicts(qd, {'Texp': tshort}))

    flat_dict = {}
    qd = {'Object': 'Twilight'}
    params = ('Filter', 'Texp')
    flatConfigs = unique(obslog.query(qd)[params])
    for config in flatConfigs:
        filt, t = config
        config_dict = dict(zip(params, config))
        lampsOn = obslog.file_query(merge_dicts(qd, config_dict))
        lampsOff = obslog.file_query({'ObsType': 'DARK', 'Texp': t})
        bpmFile = 'MCbpm_'+filt+'.pl'
        outfile = 'MCflat_'+filt
        flat_dict[outfile] = {'bpm': 'MCbpm_'+filt+'.pl',
                              'lampsOn': lampsOn, 'lampsOff': lampsOff,
                              'shortDarks': shortDarks}
     return flat_dict
```

4.4.3 Creating the flatfields

The same function is used to create the flatfields, irrespective of whether they are GCAL flats or sky flats, with the value of the boolean `gcal` parameter indicating the type of flats, since this affects the parameters for the `niflat` task.

```
def reduceFlats(flat_dict, gcal=True):
    prepPars, flatPars = get_pars('f2prepare', 'niflat')
    prepPars['fl_nonlinear'] = 'no'         # Fudge to fix (slightly)
    flatPars['key_nonlinear'] = 'SATURATI' # over-exposed flats
```

(continues on next page)

(continued from previous page)

```
    if not gcal:
        flatPars.update({'fl_rmstars': 'yes', 'scale': 'median'})
    for (outfile, bpmFile), (lampsOn, lampsOff,
                             shortDarks) in flat_dict.items():
        for f in shortDarks+lampsOn+lampsOff:
            if not os.path.exists('p'+f+'.fits'):
                f2.f2prepare(f, **prepPars)
        flatPars.update({'darks': filelist('p', shortDarks),
                         'lampsoff': filelist('p', lampsOff),
                         'flatfile': outfile, 'bpmfile': bpmFile})
        niri.niflat(filelist('p', lampsOn), **flatPars)
    iraf.imdelete('pS*.fits')
```

The default `gcal=True` assumes that the flatfields are GCAL flats, so should be combined directly without scaling; if `gcal=False`, then the images are scaled to the same median and stars are identified and removed.

The 8-second exposure time chosen for the *Ks* flats causes pixels in the bottom-left quadrant of the detector to creep into the non-linear regime, and they will therefore be flagged during preparation and flatfield creation. In practice, we do not want this to happen, so we choose not to flag non-linear pixels in **f2prepare**, and ignore them when making the flatfield by telling **niflat** that the non-linearity threshold is actually the saturation level. Once the detector properties of F2 were better determined, shorter flatfield exposures were taken and this fudge should not be needed for more recent data. Those two lines should be removed if the illumination level of your flats is within acceptable limits.

Since the short darks will be the same for all images, we check whether the prepared files are already on disk before calling **f2prepare**. The parameters for **niflat** are then updated with the lists of prepared input files and the task is executed.

4.5 Science targets

Depending on your scientific aims and observing strategy, there are many ways that the science frames could be combined; for example, you may wish to combine all the images in a given filter together into a single output image, or you may be looking for variability and so want to produce multiple output images. In addition, you may be observing a sparse field where a sky frame can be created from the observations of the target, or you may need to take sky frames at offset positions. For this reason, and to provide flexibility, the dictionary detailing how to produce the science images is constructed with the help of a *Target configuration file*.

There is a single function that constructs dictionaries for both the science output images *and* the sky frames.

```
def selectTargets(obslog):
    with open('imgTargets.yml', 'r') as yf:
        targets = yaml.load(yf)
    sci_dict = {}
    qd = {'ObsClass': 'science'}
    for outfile, pars in targets.items():
        sciFiles = obslog.file_query(merge_dicts(qd, pars))
```

(continues on next page)

(continued from previous page)

```
        t, filt = obslog[sciFiles[0]]['Texp', 'Filter']
        file_dict = {'dark': pars.get('dark', 'MCdark_'+str(int(t))),
                     'bpm': pars.get('bpm', 'MCbpm_'+filt),
                     'flat': pars.get('flat', 'MCflat_'+filt),
                     'sky': pars.get('sky', 'self')}
        try:
            groupsize = pars['groupsize']
        except:
            sci_dict[outfile] = merge_dicts(file_dict, {'input': sciFiles})
        else:
            index = 1
            while len(sciFiles) > 0:
                sci_dict['{}_{:03d}'.format(outfile, index)] = merge_dicts(
                    file_dict, {'input': sciFiles[:groupsize]})
                del sciFiles[:groupsize]
                index += 1
    # Create list of sky frames used in reduction
    sky_list = [v['sky'] for v in sci_dict.values()]
    # Make a reduction dict of bespoke skies
    sky_dict = {k: v for k, v in sci_dict.items() if k in sky_list}
    # And then remove these from the science reduction dict
    sci_dict = {k: v for k, v in sci_dict.items() if k not in sky_list}
    return sky_dict, sci_dict
```

The configuration file is opened and each entry processed by first constructing a list of input science frames using the query parameters provided in this file. It is *assumed* that these frames all have the same exposure time and filter, so these properties are determined from the first frame in the list. The exposure time is used to determine the name of the **MasterCal Dark**, and the filter is used to determine the names of the **MasterCal Flat** and **MasterCal BPM**, if these are not explicitly provided in the file.

Each entry can have an optional `sky` parameter. This can have the value `none` (indicating no sky-subtraction is required, e.g., for short-exposure standard stars) or the name of a specific file. It can be absent for two reasons: either this output image is a sky frame to be subtracted from another science frame, or a sky frame should be constructed from the same science frames. If absent, the word `self` is used as a placeholder.

In principle, we're now good to go, but it may be desirable to group the input files up into more manageable chunks. This is indicated in the *Target configuration file* by the use of a `groupsize` entry. If absent, a single entry is created in the science reduction dictionary, containing all the science input files. If this key does exist, however, then the input file list is broken up, `groupsize` files at a time, to produce separate entries in the reduction dictionary (all using the same MasterCals), until there are no files left. The output filenames are given suffixes `_001`, `_002`, etc. Grouping the images in this way copes with variations in the sky better, and also means a problem during the coaddition of the frames won't affect the entire output.

At this stage, the reduction dictionary may include bespoke sky frames, which need to be separated since they will be reduced differently (and also need to be reduced before the science frames for which they are used). A list the sky frames is extracted from the reduction dictionary, and any entries with output files matching an entry in this list are placed in their own dictionary and removed from the science dictionary. The function returns both dictionaries for processing.

4.5.1 Bespoke sky images

The construction of sky images is straightforward with the task **nisky**, which performs a two-pass procedure to mask objects from the images before combining them.

```
def reduceSkies(sky_dict):
    prepPars, redPars, skyPars = get_pars('f2prepare', 'nireduce', 'nisky')
    for outfile, file_dict in sky_dict.items():
        darkFile = file_dict['dark']
        prepPars['bpm'] = file_dict['bpm']
        flatFile = file_dict['flat']
        skyFiles = file_dict['input']
        for f in skyFiles:
            f2.f2prepare(f, **prepPars)
            gemtools.gemarith('p'+f, '-', darkFile, 'dp'+f, **arithPars)
        skyPars['outimage'] = 'nf_'+outfile
        niri.nisky(filelist('dp', skyFiles), **skyPars)
        redPars.update({'fl_dark': 'no', 'fl_flat': 'yes',
                        'flatimage': flatFile, 'outimage': outfile})
        niri.nireduce('nf_'+outfile, **redPars)
        iraf.imdelete('nf_'+outfile)
    iraf.imdelete('pS*.fits')
```

The input images have to be prepared and dark-subtracted before being sent to **nisky**. Note that the output sky frame is *not* flatfielded, so we have to flatfield it in order to ascertain how well the object masking has worked. The dark-subtracted files are deliberately not removed from disk by this function; if the sky is being constructed from dithered images of a science target, they will be used for the reduction of the science image.

4.5.2 Science images

The final science data products are constructed from the science reduction dictionary. The precise series of reduction steps depends on the manner in which sky-subtraction is to be performed (if at all) and so there are logic-dependent blocks in the tutorial code that may be irrelevant for your own dataset.

```
def reduceScience(sci_dict):
    prepPars, arithPars, redPars, coaddPars = get_pars('f2prepare', 'gemarith
↪',
                                                       'nireduce', 'imcoadd')
    for outfile, file_dict in sci_dict.items():
        darkFile = file_dict['dark']
        prepPars['bpm'] = file_dict['bpm']
        flatFile = file_dict['flat']
        skyFile = file_dict['sky']
        sciFiles = file_dict['input']
        if skyFile == 'self':
            # Make the sky (also leaves dark-subtracted files on disk)
            skyFile = outfile+'_sky'
            reduceSkies({skyFile: file_dict})
        else:
```

(continues on next page)

(continued from previous page)

```
            for f in skyFiles:
                f2.f2prepare(f, **prepPars)
                gemtools.gemarith('p'+f, '-', darkFile, 'dp'+f, **arithPars)
        redPars.update({'outprefix': 'f', 'fl_sky': 'no',
                        'flatimage': flatFile})
        niri.nireduce(filelist('dp', sciFiles), **redPars)
        imcoadd_infiles = filelist('fdp', sciFiles)
        if skyFile != 'none':
            redPars.update({'outprefix': 'r', 'fl_flat': 'no',
                            'fl_sky': 'yes', 'skyimage': skyFile})
            niri.nireduce(filelist('fdp', sciFiles), **redPars)
            imcoadd_infiles = filelist('rfdp', sciFiles)
        coaddPars.update({'badpixfile': bpmFile, 'outimage': outfile})
        gemtools.imcoadd(imcoadd_infiles, **coaddPars)
    iraf.imdelete('pS*.fits,dpS*.fits,rdpS*.fits,rfdpS*.fits')
```

Each entry in the science reduction dictionary is reduced in turn. If the sky frame is to be constructed from the science frames themselves (i.e., the entry has `self` instead of a filename), then `reduceSkies()` is called to produce that frame, which is named after the science output file but given the suffix `_sky`. This will leave the individual dark-subtracted files on disk which can be used for the next steps; otherwise, the files must be prepared and dark-subtracted.

The individual science frames are then flatfielded, and sky-subtraction takes place if requested. Note that, if no sky-subtraction is to take place (e.g., for short exposures of photometric standards) the `fl_sky` parameter for **nireduce** has to be explicitly set to `no` (even though that is the default) in case a previous iteration of the loop had changed its value. Finally, the reduced files are sent to **imcoadd** to be aligned and stacked.

> **Caution:** **imcoadd** can be quite fickle, especially when there are artifacts in the data such as caused by the peripheral wavefront sensor guide probe. It is recommended that you run the task interactively and check each coordinate fit (by pressing `x` and `y` once the interactive fitting window appears) to confirm that the points scatter around zero and have an rms of a few tenths of a pixel.

An additional function, `coaddScience()` is provided in the tutorial script. If you find you are having problems with **imcoadd**, it can be frustrating and time-consuming to have to go back to the raw data and perform the standard reduction steps again, and you may wish to fully reduce the individual files but not combine them at this stage. Simply comment out the call to **imcoadd** and remove the appropriate files from the deletion list in the next line to ensure they remain on disk. Then call `coaddScience()` with the appropriate science reduction dictionary to complete the reduction.

If you have broken your list of science files into groups then neither `reduceScience()` nor `coaddScience()` will combine these groups. You will need to make a final call to **imcoadd** with the separate stacks if you wish to have a single stacked output image.

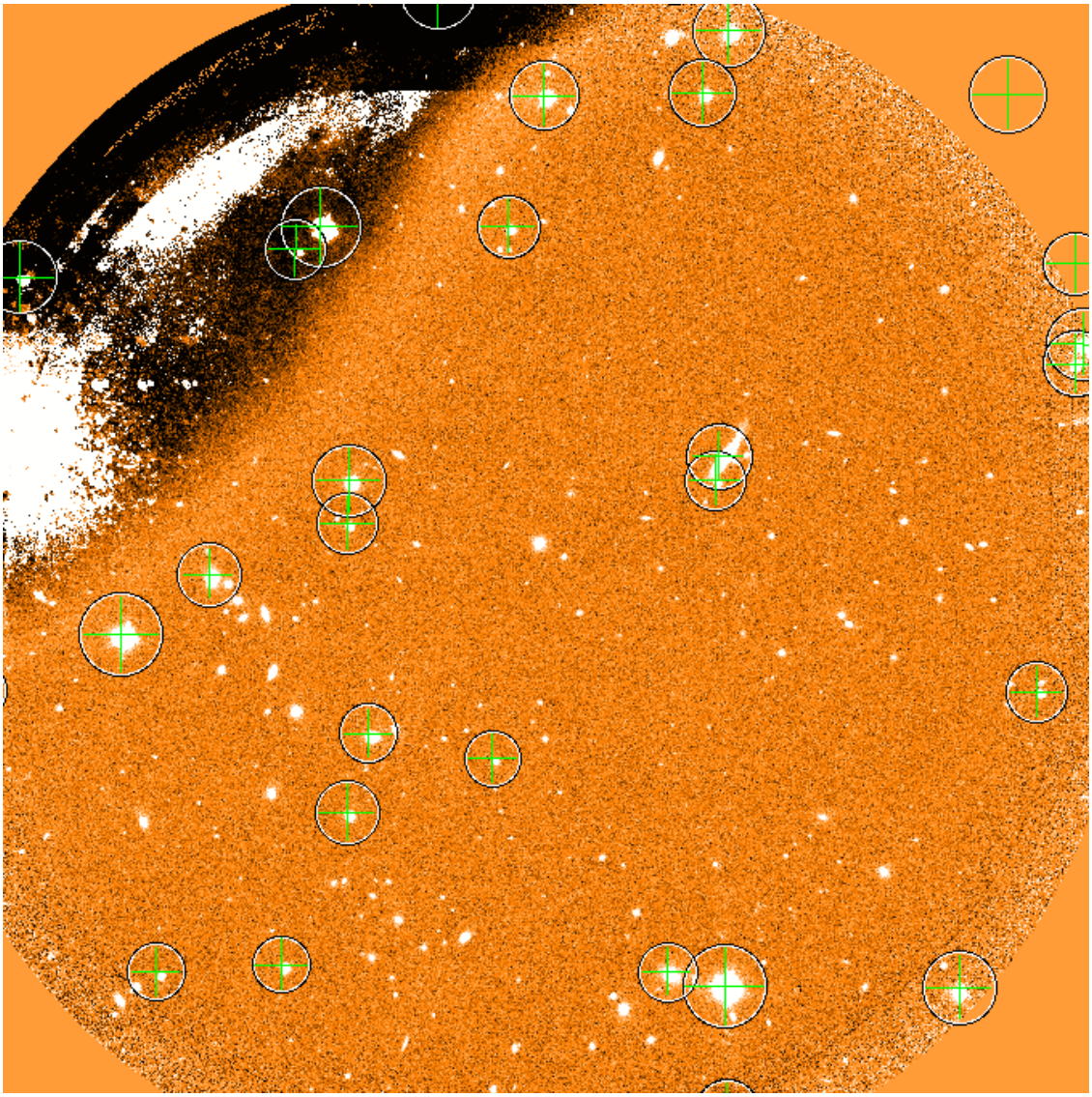

Fig. 1: Y-band image produced by this tutorial, with locations of stars from the 2MASS catalog indicated. Note the slight astrometric offset and the effect of the peripheral wavefront sensor in the top left of the image.

4.6 Astrometric calibration

The astrometry given in the image headers should be accurate enough to allow you to identify your target(s) in the image, but there may be an offset of a few arcseconds. The simplest way to correct for this offset is by using an image display tool to measure the pixel coordinates of an objects whose celestial coordinates you know (e.g., from a catalog). You can then update the world coordinate system with the following four IRAF/PyRAF commands:

```
hedit image[SCI] CRPIX1 x
hedit image[SCI] CRPIX2 y
hedit image[SCI] CRVAL1 ra
hedit image[SCI] CRVAL2 dec
```

where *x* and *y* are the pixel coordinates, and *ra* and *dec* the celestial coordinates (in degrees) of the object.

Alternatively, or if you require a more accurate WCS, you can try one of the following options:

- the IRAF MSCRED[25] package
- the astrometric calibration tools in Gaia (part of the Starlink Software[26])
- astrometry.net[27]

4.7 Flux calibration

The reduced images of your science target(s) and standard star(s) are in ADUs per exposure. After measuring the brightness of both objects in these units, the magnitude of your science target is simply given by:

$$m_{sci} = m_{std} - 2.5 \log_{10} \frac{c_{sci}/t_{sci}}{c_{std}/t_{std}}$$

where m_{std} is the magnitude of the standard star in the filter used; c_{sci} and c_{std} are the summed counts for the science target and standard, respectively; and t_{sci} and t_{std} are the exposure times of each individual frame used when observing the science target and standard.

This calculation will only be accurate if both objects were observed in clear skies (CC50) and at similar airmasses. The *atmospheric extinction* in the near-infrared is small at Cerro Pachon so a modest difference in airmasses is unlikely to affect the accuracy of the final result. If you require photometric accuracy greater than a few per cent, it is necessary to either calibrate the photometry directly from sources in the image or observe bespoke photometric standards.

[25] http://iraf.noao.edu/projects/ccdmosaic/astrometry/astrom.html
[26] http://starlink.eao.hawaii.edu/starlink/Releases
[27] http://astrometry.net

CHAPTER 5

F2 Longslit Tutorial

The recipe described here provides a recommended, **but not unique** path for processing your FLAMINGOS-2 science data.

This tutorial will use observations from program GS-2014B-Q-17 (PI:Leggett), longslit spectra of the faint Y1 star WISE J035000.32-565830.2, obtained between 20 14-Nov-07 and 2014-Dec-04. The spectra were obtained with the 4-pixel wide (0.72 arcsec) slit and the JH grism at $1.4\mu m$. Telluric standards were included in the observing plan, as were Ar comparison arcs. Leggett et al. (2016; *[L16]*) describe the data reduction procedures they followed, which are very similar to those described below.

5.1 Retrieving the Data

The first step is to retrieve the data from the Gemini Observatory Archive (see *Archive Searches*). The observations we are interested in are part of a larger program with multiple targets, so we only wish to obtain the observations in a particular date range.

```
https://archive.gemini.edu/searchform/GS-2014B-Q-17/spectroscopy/20141107-
↪20141231/
```

After retrieving the science data, click the **Load Associated Calibrations** tab on the search results page and download the associated dark and flat-field exposures. Calibrations are found many days either side of the actual science exposures and so it is worthwhile looking at the dates of the calibrations and removing some of the more distant ones (e.g., there are 6 flats from early October 2014, one from 29 Dec, and two from January 2015).

5.1.1 Exposure Summary

The data contain exposures of a specific science target, telluric standards, and dayCal calibrations that are summarized in the table below. *It is important to review the observing log database to understand how to approach the data processing.* All exposures were obtained with `ReadMode = Bright`.

Table 1: **Exposure Summmary**

Target	T_exp	N_exp	Date
WISE 0350-56	300	52	2014-Nov-07
	300	6	2014-Nov-13 (not used)
	300	16	2014-Dec-04
HD 13517 (F3V)	5	8	2014-Nov-07
HD 17813 (F2V)	5	8	2014-Nov-13 (not used)
HD 30526 (F7V)	5	4	2014-Nov-07
HD 36636 (F3V)	5	4	2014-Dec-04
Dark	300	24	2014-Nov-08 — 2014-Dec-13
	15	27	2014-Nov-08 — 2014-Dec-13
	5	27	2014-Nov-08 — 2014-Dec-13
	4	21	2014-Nov-08 — 2014-Dec-13
GCAL Flat	4	10	2014-Nov-05 — 2014-Dec-15
GCAL Ar Arc	15	3	2014-Nov-07 — 2014-Dec-04

5.2 Preparation

First download `obslog.py` to the `raw` subdirectory and create an observing log, as described in *Observing Log*.

```
python obslog.py obslog.fits
```

The other files needed for this tutorial are a python script and two configuration files.

- Download: `reduce_ls.py`

This python script will perform an automated reduction of the spectroscopy of WISE J0350; see the section *Using the Python scripts* to understand how to use it. This tutorial will take you through it, step by step, so you can see understand the procedure and how to edit it for your own F2 longslit data, should you choose to do so.

Configuration files are required for the IRAF task parameters that differ from the defaults, and to provide the script with information about the targets.

- Download IRAF task parameters: `lsTaskPars.yml`
- Download target information: `lsTargets.yml`

5.2.1 Target configuration file

In order to process the observations of celestial targets (both telluric and/or spectrophotometric standards and science targets), it is necessary to prepare a file describing the ways in which the files should be combined. For the tutorial, you should download `lsTargets.yml`

The contents of this file are as follows:

```
HD13517:
    Object:    HD 13517
    Date:      '2014-11-07'
    arc:       arc_S20141107S0235

HD30526:
    Object:    HD 30526
    Date:      '2014-11-07'
    arc:       arc_S20141107S0263

HD36636:
    Object:    HD 36636
    Date:      '2014-12-04'
    arc:       arc_S20141204S0075
    nsreduce:
        skyrange:  90

epoch1:
    first:     S20141107S0207
    last:      S20141107S0234
    arc:       arc_S20141107S0235
    telluric:  HD13517

epoch2:
    ObsID:     GS-2014B-Q-17-62
    first:     S20141107S0239
    arc:       arc_S20141107S0263
    telluric:  HD30526

epoch3:
    Date:      '2014-12-04'
    arc:       arc_S20141204S0075
    telluric:  HD36636
```

The order of entries is irrelevant, but each entry is labeled with an output filename, and information that will allow the construction of a dictionary with which to query the observation log to produce a list of input files. There are several ways to do this, including specifying the `first` and `last` filenames, or the date, or the Observation ID.

Note: If you are selecting by a single date, the date string **must** be enclosed in quoted to prevent it being parsed into a python date object.

In addition, the name of a wavelength calibration (the *output* file from **nswavelength**) *must* be provided, and

a telluric absorption standard if one is to be used. The keywords `dark`, `flat`, and `bpm` can also be used to specify additional calibration files, if the default choices are inappropriate. Finally, additional parameters for **nsreduce** and/or **nsextract** can also be given as indicated.

More details about this file are given in the sections on *Telluric standards* and *Science targets*, where it is used.

5.2.2 Configuration of nsreduce

The **nsreduce** task has several parameters; the table below lists the defaults for the processing flags — i.e., the parameters with logical values to indicate whether to perform an operation.

Table 2: **nsreduce Processing Flag Defaults**

Flag	Default	Description
`fl_cut`	Yes	Cut images using F2CUT?
`fl_corner`	Yes	Set the science arrays to zero?
`fl_process_cut`	Yes	Should cutting be performed before or after processing?
`fl_nsappwave`	Yes	Insert approximate wavelength WCS keywords into header?
`fl_dark`	No	Subtract dark image?
`fl_save_dark`	No	Save processed dark files?
`fl_sky`	No	Perform sky subtraction using skyimages?
`fl_flat`	Yes	Apply flat-field correction?
`fl_vardq`	Yes	Propagate VAR and DQ?

The parameter values need to be chosen carefully, as the order of operations performed by the task is not consistent with the order adopted in this tutorial. This means **nsreduce** will be invoked multiple times, with different flag settings, to accomplish the processing steps in the needed order.

5.3 Darks

Since dark frames are the same irrespective of whether they are used for imaging or spectroscopic observations, the procedure for reducing them is identical to that described in the Imaging Tutorials' section on *Darks*.

5.4 Flatfields

Construction of the **MasterCal Flats** follows the same pattern as the darks, first constructing a dictionary with one entry for each output image, and then processing the entries in this dictionary.

```
def selectFlats(obslog):
    flat_dict = {}
    qd = {'ObsType': 'FLAT'}
    params = ('Texp', 'Disperser')
    flatConfigs = unique(obslog.query(qd)[params])
```

(continues on next page)

(continued from previous page)

```
    for config in flatConfigs:
        t, grism = config
        config_dict = dict(zip(params, config))
        flatFiles = obslog.file_query(merge_dicts(qd, config_dict))
        outfile = 'MCflat_'+grism
        flat_dict[outfile] = {'dark': 'MCdark_'+str(int(t)),
                              'bpm': 'MCbpm_'+grism+'.pl',
                              'input': flatFiles}
    return flat_dict
```

While the only thing that matters for dark frames is the exposure time, for flatfields it is the combination of disperser (grism) and filter (the central wavelength of the F2 grisms is not configurable). However, in practice there is a natural choice of filter for each of the grisms so only this is used to identify and name the flatfields. In addition, the exposure time must be determined so the appropriate dark exposure can be subtracted (while flats do not *need* to have the same exposure time to be combined, the illumination is constant so in practice this is always the case).

A list of unique combinations of exposure time and grism are extracted from the observing log (the function `unique()` is used rather than `set()` when more than one column is used) and then these are cycled through to build a dictionary. For each output flatfield (named according to the grism used), the dictionary entry has the MasterCal dark, the *output* bad pixel mask, and the list of raw input frames.

Note: The output flatfields and BPMs are only distinguished by the name of the grism. If you were to take flats with the same grism but different filters, the software would crash as it attempts to create the same files the second time around. It is up to you to ensure that the output filenames for each set of MasterCal files are *unique*.

```
def reduceFlats(flat_dict, pars):
    prepPars, arithPars, cutPars, flatPars = get_pars('f2prepare', 'gemarith',
                                                      'f2cut', 'nsflat')
    for outfile, file_dict in flat_dict.items():
        darkFile = file_dict['dark']
        bpmFile = file_dict['bpm']
        flatFiles = file_dict['input']
        for f in flatFiles:
            f2.f2prepare(f, **prepPars)
            gemtools.gemarith('p'+f, '-', darkFile, 'dp'+f, **arithPars)
            f2.f2cut('dp'+f, **cutPars)
        flatPars.update({'flatfile': outfile, 'bpmfile': bpmFile})
        gnirs.nsflat(filelist('cdp', flatFiles), **flatPars)
    iraf.imdelete('pS*.fits,dpS*.fits,cdpS*.fits')
```

With the dictionary created, it is a simple matter to cycle through the entries. For each output flatfield, each input file is prepared by **f2prepare**, dark-subtracted by **gemarith**, and cut by **f2cut**. The flatfield is then made using **nsflat**, with the output bad pixel mask being added to the default parameters previously read in from the configuration file. Finally, intermediate files are deleted.

There is an interactive step in the creation of the flatfield, as a smooth function has to be fit to the wavelength response. A cubic spline is the best option, and the order should be chosen so that it fits the major bumps and

wiggles in the response. Since the flatfield is used for both the science exposures and the telluric standard, and then a normalized version of the standard is divided into the science data, the exact choice of spline order does not have a large effect on the final data products.

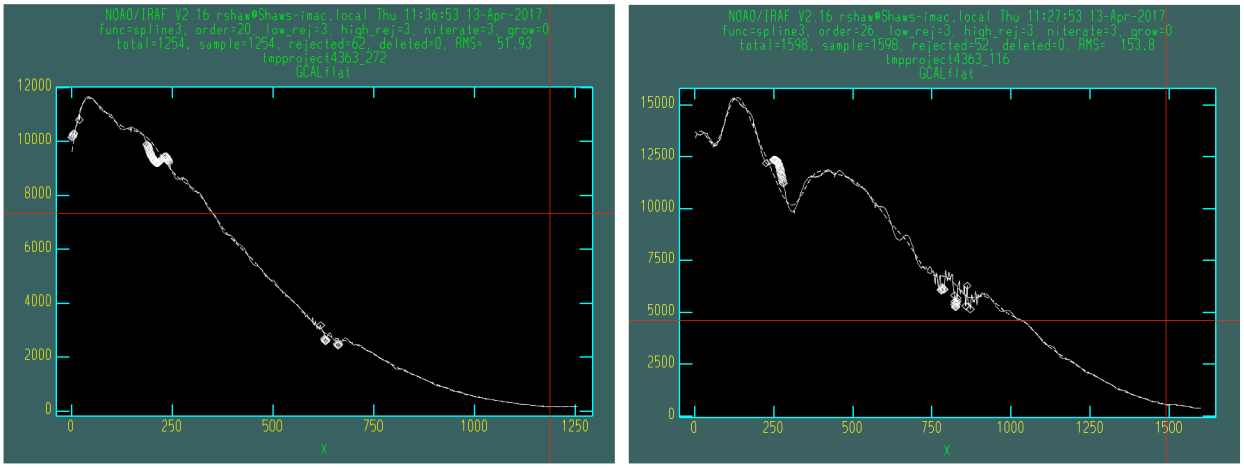

Fig. 1: Fit to the response function of the JH (*left*) and HK (*right*) combined flat-field. A `spline3` function of order 20 and 26, respectively, was used to create these normalized **Flat-field MasterCals**. Click image to enlarge.

5.5 Arcs

The wavelength calibration is derived from the spectrum of an arc lamp. It is common practice to take multiple arcs in the same configuration throughout an observing sequence and use the one taken closest in time to calibrate a particular frame. For this reason, the raw arc frames are not combined, even when they have the same configuration.

```
def selectArcs(obslog):
    arc_dict = {}
    arcFiles = obslog.file_query({'ObsType': 'ARC'})
    params = ('Texp', 'Disperser')
    for f in arcFiles:
        t, grism = obslog[f][params]
        outfile = 'arc_'+f
        arc_dict[outfile] = {'dark': 'MCdark_'+str(int(t)),
                             'flat': 'MCflat_'+grism,
                             'bpm': 'MCbpm_'+grism,
                             'input': [f]}
    return arc_dict
```

The selection of files for **Wavelength MasterCal** files is similar to that for flatfields. Each frame needs to be associated with a dark frame (of the same exposure time) and ideally a BPM file and flatfield (created using the same grism); although these are not strictly necessary, including them does improve the quality of the wavelength solution. In order to ensure the output filenames are unique, they are constructed by prepending `arc_` to the raw input filename.

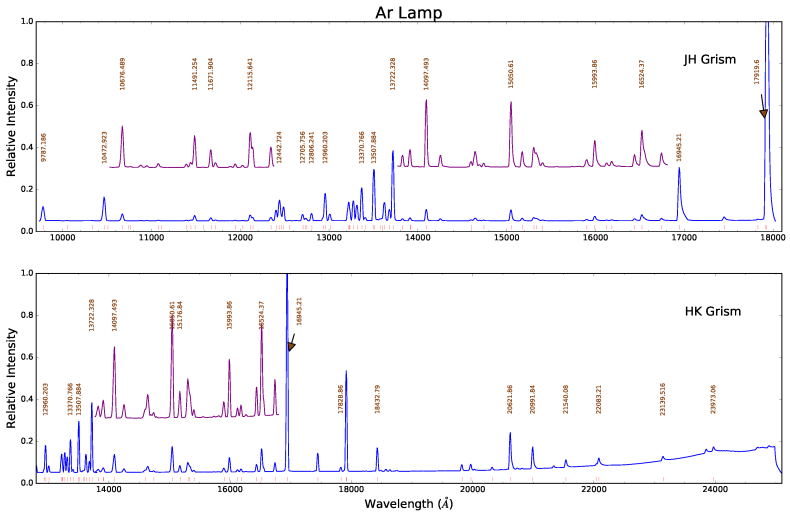

Fig. 2: Ar spectra in the *JH*- band (*upper*) and *HK*- band (*lower*) at full scale (*blue*), with portions magnified (*purple*) and offset vertically for clarity. More than 100 identifiable lines are marked (*red ticks*) along the wavelength axis. Some of the brighter or more isolated lines are labeled, which should suffice to bootstrap a wavelength solution. Click image to enlarge.

```
def reduceArcs(arc_dict, pars):
    prepPars, arithPars, redPars, wavePars = get_pars('f2prepare', 'gemarith',
                                                      'nsreduce', 'nswavelength
↪')
    for outfile, file_dict in arc_dict.items():
        darkFile = file_dict['dark']
        prepPars['bpm'] = file_dict['bpm']
        flatFile = file_dict['flat']
        arcFiles = file_dict['input']
        for f in arcFiles:
            f2.f2prepare(f, **prepPars)
            gemtools.gemarith('p'+f, '-', darkFile, 'dp'+f, **arithPars)
        if flatFile:
            redPars.update({'fl_flat': 'yes', 'flatimage': flatFile})
        else:
            redPars['fl_flat'] = 'no'
        gnirs.nsreduce(filelist('dp', arcFiles), **redPars)
        if len(arcFiles) > 1:
            gemcombine(filelist('rdp', arcFiles), 'tmp_'+outfile, **arithPars)
            gnirs.nswavelength('tmp_'+outfile, outspectra=outfile, **wavePars)
        else:
            gnirs.nswavelength('rdp'+arcFiles[0], outspectra=outfile,
                               **wavePars)
    iraf.imdelete('*pS*.fits')
```

Each entry in the previously-created dictionary is taken in turn. The input files are prepared using the associated bad pixel mask and then dark-subtracted. A call to **nsreduce** cuts and flatfields the frame (if a flatfield is provided). At this stage, if there are multiple input files, they are combined, before **nswavelength** is called to determine the wavelength solution. Intermediate files are deleted.

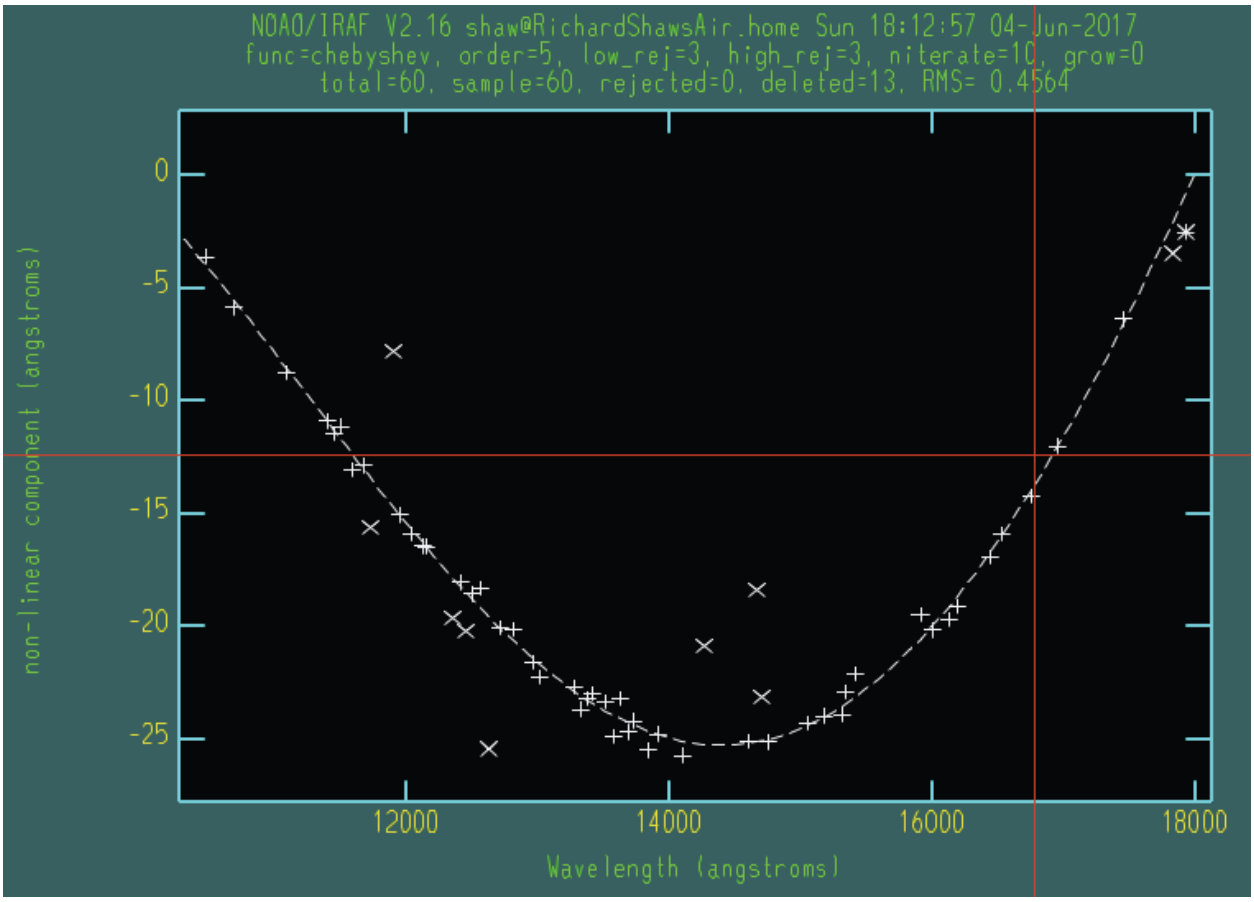

Fig. 3: Non-linear portion of a low-order fit to the dispersion for a JH arc spectrum. A `chebyshev` function of order 5 should suffice to yield an RMS < 0.5. Click image to enlarge.

Due to the fixed nature of the F2 grisms, the initial wavelength solution put in the headers should be sufficiently accurate to allow IRAF to correctly identify most of the individual arc lines and it should be possible to move directly to fitting the wavelength solution, by typing `f`. Should the initial line identification have resulted in misidentifications, these can be deleted by moving the cursor to them and typing `d`. It will then be necessary to provide the correct identifications for some lines by typing `m` to mark an identification and entering the wavelength (in Angstroms). You should then fit a solution and can type `l` to have IRAF try to identify additional lines based on this new solution. More information can be obtained by typing `?` and through the help pages for **nswavelength** and **autoidentify**. Like many of the IRAF tasks, the fitting is performed with the **icfit** task[28]. Any erroneous line identifications can be deleted interactively and an appropriate order fit applied to the data. What is considered an acceptable fit depends on the grism and your scientific goals, but an rms of 0.5 Angstroms is sufficient for this tutorial.

Having determined a suitable wavelength solution, type `q` to quit the function fitting. The **nswavelength** task will now attempt to trace the arc lines along the full length of the slit, fitting a new wavelength solution at regularly-spaced intervals in order to create a distortion map across the entire image. Although you are given the option of fitting the dispersion function interactively at each stage, this is both dull and unnecessary. At the first such prompt, type `NO` (NB. capital letters) and you will not be asked again.

5.6 Telluric standards

The lists of telluric standards and science frames are extracted at the same time from the *Target configuration file*. Each reduced telluric spectrum requires a bad pixel mask, a dark frame, a flatfield, and a wavelength calibration; each reduced science spectrum requires all of those *and* a telluric calibration. The filenames of most of these calibrations can be determined from the properties of the input frames, but the wavelength calibration and telluric standard need to be explicitly provided.

```
def selectTargets(obslog):
    with open('lsTargets.yml', 'r') as yf:
        config = yaml.load(yf)
    std_dict = {}
    sci_dict = {}
    qd = {'ObsType': 'OBJECT'}
    for outfile, pars in config.items():
        infiles = obslog.file_query(merge_dicts(qd, pars))
        t, grism = obslog[infiles[0]]['Texp', 'Disperser']
        file_dict = {'dark': pars.get('dark', 'MCdark_'+str(int(t))),
                     'bpm': pars.get('bpm', 'MCbpm_'+grism),
                     'flat': pars.get('flat', 'MCflat_'+grism),
                     'arc': pars['arc'],  # Must be specified
                     'input': infiles}
        try:
            telFile = pars['telluric']
        except KeyError:  # No telluric => treat as a standard
            std_dict[outfile] = file_dict
        else:
            sci_dict[outfile] = merge_dicts(file_dict, {'telluric': telFile})
    return std_dict, sci_dict
```

[28] http://iraf.net/irafhelp.php?val=xtools.icfit&help=Help+Page

There is one entry in the target configuration file for each output spectrum, which provides sufficient information to select the appropriate raw input files. The names of the dark, flatfield, and bad pixel mask can be provided in this file but, if absent, are constructed from the exposure time or grism. An arc must be specified. If the name of a telluric standard is not provided, then this observation is assumed to be a telluric standard itself, and it is added to the standard star reduction dictionary; otherwise it is added to the science reduction dictionary.

```
def reduceStandards(std_dict, pars):
    (prepPars, arithPars, fitcooPars, transPars, extrPars, redPars,
    combPars) = get_pars('f2prepare', 'gemarith', 'nsfitcoords',
                         'nstransform', 'nsextract', 'nsreduce', 'nscombine')
    redPars['fl_sky'] = 'yes'
    combPars['fl_cross'] = 'yes'
    with open('lsTargets.yml', 'r') as yf:
        config = yaml.load(yf)
    for outfile, file_dict in std_dict.items():
        darkFile = file_dict['dark']
        prepPars['bpm'] = file_dict['bpm']
        flatFile = file_dict['flat']
        arcFile = file_dict['arc']
        stdFiles = file_dict['input']
        for f in stdFiles:
            f2.f2prepare(f, **prepPars)
            gemtools.gemarith('p'+f, '-', darkFile, 'dp'+f, **arithPars)
        pars = merge_dicts(redPars, config[outfile].get('nsreduce', {}))
        gnirs.nsreduce(filelist('dp', stdFiles), flatimage=flatFile, **pars)
        gnirs.nscombine(filelist('rdp', stdFiles), output=outfile, **combPars)
        gnirs.nsfitcoords(outfile, lamptransf=arcFile, **fitcooPars)
        gnirs.nstransform('f'+outfile, **transPars)
        gnirs.nsextract('tf'+outfile, **extrPars)
        iraf.imdelete('f'+outfile+',tf'+outfile)
    iraf.imdelete('pS*.fits,dpS*.fits,rdpS*.fits')
```

Some of the default parameters are changed for the reduction of the standards: in particular, the individual spectra are aligned by cross-correlation before combining since the high signal-to-noise ratios makes this more accurate than using the telescope offsets.

Note: Since science targets without telluric standards have the same reduction steps as telluric standards, they are also reduced by this function. If they are too faint for cross-correlation to work, you will need to unset this flag for those objects.

The target configuration file is reopened so that any parameters required for the successful operation of **nsreduce** can be applied. In general, such parameters should not be required; however the interval between exposure start times for the star HD 36636 is somewhat irregular and it is necessary to set the `skyrange` interval for a successful reduction.

After alignment and combining, the image is transformed so that lines of constant wavelength are perfectly horizontal across the image. The task **nsfitcoords** fits a function to the map of the arc line spectra, and **nstransform** applies this to the image. There is an (optionally) interactive step to fit a 2D polynomial to the grid of datapoints, for which modest orders should suffice, especially if your science target is a point source.

The spectrum is then extracted by **nsextract**, which can be performed interactively. Since there will be a single bright source in the slit, the automated aperture finding and resizing routines will work well and the important step is tracing the spectrum along the wavelength direction. It may be necessary to delete points at each end (by moving to them with the cursor and pressing `d`) where the signal-to-noise ratio is low, to ensure that the trace is not pulled off course.

5.7 Science targets

The dictionary to reduce the science images was constructed at the same time as the one for telluric standards and the first few reduction steps are identical, except that cross-correlation is not used to align the individual spectra. You can edit the code or the `yaml` parameter file if your targets are bright enough and you wish to use this method of alignment.

```
def reduceScience(sci_dict):
    (prepPars, arithPars, fitcooPars, transPars, extrPars, redPars, combPars,
     telPars) = get_pars('f2prepare', 'gemarith', 'nsfitcoords', 'nstransform
↪',
                         'nsextract', 'nsreduce', 'nscombine', 'nstelluric')
    redPars['fl_sky'] = 'yes'
    with open('lsTargets.yml', 'r') as yf:
        config = yaml.load(yf)
    for outfile, file_dict in sci_dict.items():
        darkFile = file_dict['dark']
        prepPars['bpm'] = file_dict['bpm']
        flatFile = file_dict['flat']
        arcFile = file_dict['arc']
        telFile = file_dict['telluric']
        sciFiles = file_dict['input']
        for f in sciFiles:
            f2.f2prepare(f, **prepPars)
            gemtools.gemarith('p'+f, '-', darkFile, 'dp'+f, **arithPars)
        pars = merge_dicts(redPars, config[outfile].get('nsreduce', {}))
        gnirs.nsreduce(filelist('dp', sciFiles), flatimage=flatFile, **pars)
        gnirs.nscombine(filelist('rdp', sciFiles), output=outfile, **combPars)
        #gnirs.nsfitcoords(outfile, lamptransf=arcFile, **nsfitcrdPars)
        apply_fitcoords(outfile, telFile, fitcooPars['database'])
        gnirs.nstransform('f'+outfile, **transPars)
        pars = merge_dicts(extrPars, config[outfile].get('nsextract', {}))
        gnirs.nsextract('tf'+outfile, **pars)
        gnirs.nstelluric('xtf'+outfile, 'xtf'+telFile, **telPars)
        iraf.imdelete('f'+outfile+',tf'+outfile)
    iraf.imdelete('pS*.fits,dpS*.fits,rdpS*.fits')
```

To provide the best correction or atmospheric absorption features, it is important that the wavelength solutions of your science target and telluric standard are identical. Rather than re-running **nsfitcoords** and remembering to use exactly the same parameters as you used for the standard, the tutorial code includes a python function, called `apply_fitcoords()` that adds the necessary header keywords to the science image, allowing **nstransform** to be run with the same solution.

Once again, **nsextract** is used to produce a one-dimensional spectrum, and target-specific parameters can be

provided in the target configuration file. Important parameters for this task include `nsum` and `line`, which indicate how many rows of the image to stack, and where, in order to provide a spatial profile of the slit to allow the target location(s) to be found. This tutorial involves spectroscopic observations of a brown dwarf, whose flux is heavily suppressed in substantial regions of the spectral coverage. The parameter `trace` is also useful if your target is faint and cannot be traced along the full wavelength range; you can set its value to the filename of a previously-extracted spectrum with a higher signal-to-noise ratio and that spectrum's trace will be used to extract the target.

5.8 Flux calibration

In principle, spectrophotometric calibration requires an observation of a source whose spectrum is known; however, the near-infrared spectra of stars of any given spectral type are similar enough that the telluric standard is acceptable for this task. You can find out the spectral type and apparent magnitudes of any named star using the SIMBAD search[29]; for example, HD 36636 is an F3V star, with accurate *JHK* magnitudes listed. Its spectrum will therefore be very similar to that of the F3V star HD 26015 listed in the IRTF Spectral Library[30] (*[IRTF]*). The spectra of stars in the near-infrared are also fairly well-modeled by blackbody spectra, and Gemini provides a list of effective temperatures[31], from which we deduce that HD 36636 can be represented as a blackbody with a temperature of 6680K.

There are several paths to obtaining a flux-calibrated spectrum of your target using the telluric standard, but here is one route:

1. Correct the telluric standard for atmospheric absorption in the same way as the science target.
2. Obtain/synthesize a model spectrum of the telluric standard with the same wavelength solution as your target.
3. Determine the *sensitivity function* (i.e., the relationship between spectral counts and flux density as a function of wavelength) using the model spectrum and the telluric-corrected standard.
4. Apply the sensitivity function to the science target, remembering to correct for the different exposure times.

These steps can all be performed with various IRAF tasks but it's not very elegant, and there are issues interfacing the fitting task from core IRAF (**continuum**) that uses simple-FITS files with Gemini-IRAF tasks that use multi-extension FITS. The tutorial code includes a function, `fluxCalibrate()`, that performs the above steps. This requires the root filenames of the science file and telluric standard, plus a model spectrum of the standard. This can be called in one of two ways:

```
# Use a spectrum downloaded from the IRTF Spectral Library
fluxCalibrate('science', 'HD36636', spectrum='F3V_HD26015.txt', hmag=8.633)
# Use a blackbody spectrum
fluxCalibrate('science', 'HD36636', teff=6680, hmag=8.633)
```

A magnitude need not be specified if a flux-calibrated spectrum is used, but any of `jmag`, `hmag`, or `kmag` can be provided, as long as the spectrum covers that bandpass.

[29] http://simbad.u-strasbg.fr/simbad/sim-fid
[30] http://irtfweb.ifa.hawaii.edu/~spex/IRTF_Spectral_Library
[31] http://www.gemini.edu/sciops/instruments/nir/photometry/temps_colors.txt

CHAPTER 6

F2 MOS Tutorial

The Multi-Object Spectrocopy (MOS) mode for F2 is being commissioned. A data reduction tutorial will be provided here once this mode is available to the community.

CHAPTER 7

Useful Resources

Resources for your data reduction needs are collected here for easy reference. Information about the FLAMINGOS-2 instrument design, operation, and observing may be found on the FLAMINGOS-2 Home Page[32].

7.1 FLAMINGOS-2 Help

For help with FLAMINGOS-2 data reduction you may find the Data Reduction Support page[33] useful. To address problems with the Gemini software, direct your inquiry as follows:

- Astroconda[34] installation and PyRAF: help@stsci.edu
- Gemini Helpdesk (http://www.gemini.edu/sciops/helpdesk/) for problems with:
  - IRAF tools, including the **gemini** packages
  - Gemini Observatory Archive
- Gemini/IRAF FAQ list[35]
- List of known problems[36] with **gemini** package software or the data.
- Gemini Data Reduction User Forum[37], a place for trading ideas, scripts, and best practices.

[32] http://www.gemini.edu/sciops/instruments/flamingos2/
[33] http://www.gemini.edu/sciops/data-and-results/data-reduction-support
[34] http://astroconda.readthedocs.io/en/latest/
[35] http://www.gemini.edu/sciops/data-and-results/data-reduction-support/faq
[36] http://www.gemini.edu/sciops/data-and-results/processing-software/data-reduction-support/known-problems
[37] http://drforum.gemini.edu/topic-tag/FLAMINGOS-2/

7.2 Standards and Catalogs

The following catalogs and other reference material may be of use during your data reduction and analysis:

- Telluric standard star compendia and a model atmosphere library
  - Spectroscopic/telluric standards[38] from Gemini
  - Standard star spectral library[39] from ESO
  - Standard star calibration library[40] from STScI
  - IRTF Spectral Library[41]
- DSS[42]: ESO Online Digitized Sky Survey
- USNO[43] Image and Catalog Archive Server

7.3 IRAF Reduction Tools

A few of the IRAF data reduction tasks have interactive options, where the user provides input via the IRAF graphics utility. These tools involve cursor interactions and keystrokes, which can be viewed by entering `?` when in cursor mode. The most commonly used options for two of the most complex tasks are given below, for reference. More detailed information can be found in the IRAF Spectroscopy Documentation[44]

7.3.1 Wavelength Calibration

Wavelength calibration is performed with the **gnirs.nswavelength** task, which is really a wrapper for the IRAF **identify** family of tasks. The following tables summarize the most common commands for this task.

[38] http://www.gemini.edu/sciops/instruments/nearir-resources/spectroscopic-standards
[39] https://www.eso.org/sci/observing/tools/standards/IR_spectral_library.html
[40] http://www.stsci.edu/hst/observatory/crds/calspec.html
[41] http://irtfweb.ifa.hawaii.edu/~spex/IRTF_Spectral_Library/
[42] http://archive.eso.org/dss/dss
[43] http://www.usno.navy.mil/USNO/astrometry/optical-IR-prod/icas
[44] http://iraf.noao.edu/docs/spectra.html

Cursor Keys

Table 1: **Identify Cursor Keys**

Key	Description
?	Clear the screen and print a menu of options.
a	Apply next *center* or *delete* operation to *all* features
b	Identify features and find a dispersion function automatically using the coordinate line list and approximate values for the dispersion.
c	Center the feature nearest the cursor. Used when changing the position finding parameters or when features are defined from a previous feature list.
d	Delete the feature nearest the cursor. Delete all features when preceded by the a ll key. This does not affect the dispersion function.
e	Find features from a coordinate list without doing any fitting. This is like the l key without any fitting.
f	Fit a function of the pixel coordinates to the user coordinates. **This enters the interactive function fitting package.**
g	Fit a zero point shift to the user coordinates by minimizing the difference between the user and fitted coordinates. The coordinate function is not changed.
i	Initialize (delete features and coordinate fit).
l	Locate features in the coordinate list. A coordinate function must be defined or at least two features must have user coordinates from which a coordinate function can be determined. If there are features an initial fit is done; then features are added from the coordinate list; and then a final fit is done.
m	Mark a new feature using the cursor position as the initial position estimate.
n	Move the cursor or zoom window to the next feature (same as +).
p	Pan to the original window after zooming on a feature.
q	Quit and continue with next image.
r	Redraw the graph.
s	Shift the fit coordinates relative to the pixel coordinates. The user specifies the desired fit coordinate at the position of the cursor and a zero point shift to the fit coordinates is applied. If features are defined then they are recentered and the shift is the average shift. The shift is printed in pixels & user coordinates & z (fractional shift).
u	Enter a new user coordinate for the current feature. When marking a new feature the user coordinate is also requested.
w	Window the graph. A window prompt is given and a number of windowing options may be given. For more help type ? to the window prompt or see help under gtools.
x	Find a zero point shift for the current dispersion function. This is used by starting with the dispersion solution and features from a different spectrum. The mean shift is printed in user coordinates & mean shift in pixels & the fractional shift in user coordinates.
z	Zoom on the feature nearest the cursor. The width of the zoom window is determined by the parameter zwidth.
.	Move the cursor or zoom window to the feature nearest the cursor.
+	Move the cursor or zoom window to the next feature.
−	Move the cursor or zoom window to the previous feature.

Colon-command Summary

The following is an abridged list of *colon commands* (i.e., command names preceded by the `:` key) to view (with no argument) or set (including trailing argument) a **nswavelength** task parameter. The commands may be abbreviated. For a full list see identify[45] or invoke the `?` cursor command within an interactive session.

Table 2: **Identify Colon Commands**

Key	Value	Description
`:show`	file	Show the values of all the parameters. If a file name is given then the output is appended to that file. If no file is given then the terminal is cleared and the output is sent to the terminal.
`:features`	file	Print the feature list and the fit rms. If a file name is given then the output is appended to that file. If no file is given then the terminal is cleared and the output is sent to the terminal.
`:coordlist`	file	Set or show the coordinate list file.
`:cradius`	value	Set or show the centering radius in pixels.
`:threshold`	value	Set or show the detection threshold for centering.
`:database`	name	Set or show the database for recording feature records.
`:ftype`	value	Set or show the feature type (emission or absorption).
`:fwidth`	value	Set or show the feature width in pixels.
`:labels`	value	Set or show the feature label type (`none|index|pixel|coord|user|both`). None produces no labeling; index labels the features sequentially in order of pixel position; pixel labels the features by their pixel coordinates; coord labels the features by their user coordinates (such as wavelength); user labels the features by the user or line list supplied string; and both labels the features by both the user coordinates and user strings.
`:match`	value	Set or show the coordinate list matching distance.
`:maxfeatures`	value	Set or show the maximum number of features automatically found.
`:minsep`	value	Set or show the minimum separation allowed between features.
`:zwidth`	value	Set or show the zoom width in user units.

7.3.2 APEXTRACT Summary

The aperture extraction utility (apextract[46]) in IRAF is invoked from the **gnirs.nsextract** task. When run interactively, this utility provides a variety of cursor keys to control the extraction of target spectra. If you use IRAF for your data reduction, you will need to get comfortable with this task.

Cursor Keys

The following are the available cursor commands for aperture definition and spectrum extraction.

[45] http://stsdas.stsci.edu/cgi-bin/gethelp.cgi?identify
[46] http://stsdas.stsci.edu/cgi-bin/gethelp.cgi?apextract.men

Table 3: **Aperture Editor Cursor Keys**

Key	Ap	Description
?		Print help
a		Toggle the ALL flag
b	an	Set background fitting parameters
c	an	Center aperture(s)
d	an	Delete aperture(s)
f		Find apertures up to the requested number
g	an	Recenter aperture(s)
l	ac	Set *lower* limit of current aperture at cursor position (see u)
m		Define and center a new aperture on the profile near the cursor
n		Define a new aperture centered at the cursor
q		Quit
r		Redraw the graph
s	an	Shift the center(s) of the current aperture to the cursor position
t	ac	Trace aperture positions
u	ac	Set *upper* limit of current aperture at cursor position (see l)
w		Window the graph using the window cursor keys
y	an	Set aperture limits to intercept the data at the cursor y position
z	an	Resize aperture(s)
.	n	Select the aperture nearest the cursor to be the current aperture
+	c	Select the next aperture (in ID) to be the current aperture
−	c	Select the previous aperture (in ID) to be the current aperture
I		Interrupt task immediately. Database information is not saved.

The letter a following the key indicates if all apertures are affected when the ALL flag is set. The letter c indicates that the key affects the *current* aperture while the letter n indicates that the key affects the aperture whose center is *nearest* the cursor.

Colon-command Summary

The following is an abridged list of colon commands (i.e., command names preceded by the : key) to view (with no argument) or set (including trailing argument) a **nsextract** task parameter. For a full list see apall[47] or invoke the ? cursor command within an interactive session.

[47] http://stsdas.stsci.edu/cgi-bin/gethelp.cgi?apall

Table 4: **Aperture Editor General Colon-commands**

Command	Description
`:b_function`	Background fitting function
`:b_function`	Background fitting function
`:b_high_reject/ :b_low_reject`	Background high/low rejection limits
`:b_naverage`	Determine background from average or median
`:b_order`	Function order for background fit
`:b_sample`	Comma-separated list of background sample region(s) [`nnn:nnn`]
`:background`	Background to subtract (e.g. `none`)
`:bkg`	Subtract background in automatic width? [`yes` | `no`]
`:clean`	Detect and replace bad pixels? [`yes` | `no`]
`:extras`	Extract sky & sigma etc. in addition to spectrum?
`:line`	Dispersion line over which to display profile
`:nsum`	Extent over which to determine profile (positive for *sum* or negative for *median*)
`:lower/:upper`	Lower/upper aperture limits relative to center
`:lsigma/:usigma`	Lower/upper rejection threshold
`:parameters`	Print the current value of all parameters
`:radius`	Profile centering radius
`:t_function`	Type of fitting function for trace
`:t_high_reject/ :t_low_reject`	Upper/lower rejection limits for trace [sigma]
`:t_nsum`	Number of dispersion pixels to sum for trace
`:t_order`	Order of trace fitting function
`:t_step`	Step size for fitting function
`:weights`	Extraction weights [`none` | `variance`]
`:width`	Profile centering width

Note that all parameters having to do with positions or distances are in units of pixels.

7.4 Acknowledgements

Scientific publications that make use of data obtained from Gemini facilities should include the appropriate acknowledgement described on the Gemini Observatory Acknowledgements[48] page. You should also cite the FLAMINGOS-2 instrument description paper by *[EE]*.

Citations to this *Cookbook* should read:

> Shaw, R. A. and Simpson, C. 2018, *FLAMINGOS-2 Data Reduction Cookbook* (Version 1.1; Hilo, HI: Gemini Observatory)

Use of IRAF software should include the following footnote:

[48] http://www.gemini.edu/sciops/data-and-results/acknowledging-gemini

> *IRAF is distributed by the National Optical Astronomy Observatory, which is operated by the Association of Universities for Research in Astronomy (AURA) under a cooperative agreement with the National Science Foundation.*

Use of PyRAF software should also include the following footnote:

> *PyRAF is a product of the Space Telescope Science Institute, which is operated by AURA for NASA.*

CHAPTER 8

Glossary

ADU (Analog-to-Digital Unit) For sensor pixel arrays used in Optical/IR astronomy, one *analog-to-digital unit* is one count, corresponding to a quantity of detected photons given by the detector gain setting.

Baseline Calibrations *Baseline calibration* exposures are those that are obtained by Gemini Staff throughout an observing semester to support the science programs that have been initiated. They include closed-dome (bias, dark, flat-fields) and on-sky (standard stars) observations. See NIR Baseline Calibrations[49] for details.

BPM (Bad Pixel Mask) A static *Bad Pixel Mask* can be defined for each focal plane, which consists of an array of short integers that encode pathologies, such as bad and saturated pixels. The BPM is inserted into the *MEF* files as a DQ extension.

CDS (Correlated Double-Sample) A method of reading IR sensors in which one or more non-destructive read pairs are obtained just before and just after the exposure. Since there is no shutter for IR cameras, this method provides a means of correcting for the counts that accumulate before the exposure starts, which can be quite substantial in the K-band thermal IR.

Darks *Dark* exposures are of finite duration taken with the shutter closed. They are used to characterize the background that arises from the sensor array, some (or in the *K*– band, most) of which is thermal in origin. In the IR, it is essential that the duration of the dark exposures match that of the science or calibration exposures that are being corrected.

dayCal A variety of calibration exposures are obtained during the day (or twilight) to support observing programs, and to monitor the health and performance of GMOS.

Dither A *dither* pattern of exposures in a sequence, with small relative spatial offsets. Dither may be used to provide full coverage of a contiguous region of sky or, in the IR, enable the creation of a sky frame (i.e., with sources excluded). See, e.g., *Night-sky frames*.

[49] http://www.gemini.edu/sciops/instruments/nearir-resources/baseline-calibrations

FITS The *Flexible Image Transport System* format is an international standard in astronomy for storing images, tables, and related metadata in disk files. Multiple images and tables may be stored in *MEF* files. See the IAU FITS Standard[50] *[FITS]* for details.

FoV The *field of view*, or spatial extent of sky the from the optical system of the telescope+instrument that actually falls on the detector.

Gain The conversion factor between electrons and ADU when the detector is read out. A gain of 10 means that it requires 10 electrons to produce one ADU.

GCAL The Gemini Facility Calibration Unit[51] provides continuum and emission light sources for flat-field and wavelength calibration of various instruments.

HDU A *FITS* file consists of a primary *header-data unit* and zero or more extension HDUs. The primary HDU (*PHU*) contains a header, but may or may not contain an image. An extension HDU contains a header and any valid FITS extension type, including a binary image or a table.

Keyword In the astronomy data domain, *keyword* is most closely associated with a named metadatum stored in a *FITS* header, which is assigned a particular scalar or text value represented as ASCII.

Lamps-Off A *GCAL* exposure taken with the flat-field lamp turned *off*. The signal from such exposures taken in the IR consists of a thermal component from the instrument, which may be subtracted from a combined *Lamps-On* exposure. This signal is quite strong in *K*– band, so there is no need to obtain separate Lamps-On exposures.

Lamps-On A *GCAL* exposure taken with the flat-field lamp turned *on*. The signal from such exposures taken in the IR will include a thermal component from the instrument, which is measured with *Lamps-Off* exposures.

MasterCal A *Master Calibration* file, which may be an image that captures a particular instrumental signature, or a table consisting of a calibration or reference information. *MasterCals* are often built by combining calibration exposures in a particular way, or by recording coefficients of a function that characterizes a calibration, e.g., the dispersion solution or a linearity correction. They may also consist of a catalog of reference information, such as astrometric or photometric standards.

MDF (Mask Definition File) The *mask definition file* is a *FITS* binary table that gives the attributes of all apertures for the mask in use during the observation.

MEF *Multi-extension FITS* format files contain a primary *HDU*, and one or more extensions HDUs, each of which contains a header and data such as a table or binary image. Raw exposures from Gemini normally contain a *PHU* with no associated data array, and one image extension for each amplifier used for read-out.

MOS *Multi-object spectroscopy* is the capability of obtaining multiple spectra of different targets (or different regions within an extended object) in the same exposure. This may be achieved by orienting a facility longslit to include more than one target, or using a custom slitmask with multiple apertures corresponding to the targets of interest within the *FoV*.

OIWFS The *On-Instrument Wavefront Sensor* provides guiding and tip-tilt correction information to the telescope control system. It is mounted on a probe that patrols an area that can extend into the imaging FoV.

[50] http://fits.gsfc.nasa.gov/fits_standard.html
[51] http://www.gemini.edu/sciops/telescopes-and-sites/calibration-gcal

PHU Primary header-data unit of a *FITS* file, i.e., extension [0] in an *MEF* file. A simple FITS file contains a header and (usually) an image array. The PHUs in *MEF* files generally *do not* include an image array.

Slit-mask A *slit-mask* may be inserted at the focal-plane entrance to an instrument to block light from all except selected targets or regions within an extended source from passing through the instrument. Small apertures are cut into the mask that correspond spatially with targets of interest. The apertures are usually square (for alignment on field stars) or rectangular (i.e., *slitlets*), with the long axis sized to capture the target and nearby background, without overlapping spatially with other slitlets as projected on the detector.

WCS (World Coordinate System) A mapping from image pixel coordinates to physical coordinates. For direct images the mapping is to the equatorial (RA, Dec) system; for extracted spectra the mapping is to the dispersion axis, usually in Angstroms, and position along the slit.

Bibliography

[EE] Eikenberry, S., et al. 2004, *FLAMINGOS-2: the facility near-infrared wide-field imager and multi-object spectrograph for Gemini*, SPIE, 5492, 1196

[EE3] Eikenberry, S., et al. 2008, *FLAMINGOS-2: the facility near-infrared wide-field imager and multi-object spectrograph for Gemini*, SPIE, 7014, 0V

[L15] Leggett, et al. 2015, *Near-infrared Photometry of Y Dwarfs: Low Ammonia Abundance and the Onset of Water Clouds*, ApJ, 799, 37

[L16] Leggett, et al. 2016, *Near-IR Spectroscopy of the Y0 WISEP J173835.52+273258.9 and the Y1 WISE J035000.32-565830.2: The Importance of Non-Equilibrium Chemistry*, ApJ, 824, 2

[FITS] Pence, W.D., Chiappetti, L, Page, C.G., Shaw, R.A., & Stobie, E. 2010, *Definition of the Flexible Image Transport System (FITS), Version 3.0*, A&A, 524, 42

[IRTF] Rayner, J. T., Cushing, M.C., & Vacca, W.D. 2009, *The Infrared Telescope Facility (IRTF) Spectral Library: Cool Stars*, ApJS, 185, 289