Tutorial Series - GNIRS Imaging Data Reduction with DRAGONS Documentation

Release 3.1.0-dev

Kathleen Labrie

December 2021

Contents:

Document ID

PIPE-USER-117_GNIRSImg_DRtutorial

CHAPTER 1

Overview

This is a collection of tutorials for the reduction GNIRS keyhole imaging data with DRAGONS.

GNIRS is a spectrograph. It uses a keyhole to image a small area of the sky and acquire targets. The use of the GNIRS keyhole is not recommended for imaging but it has been used in that capacity in the past when there were no other options immediately available. The quality of the data, and therefore of the reduction is somewhat unpredictable. The quality of the reduction critically depends on the quality and the availability of the calibration frames, the flat fields in particular.

In here are tutorials that you, the reader, can run and experiment with. This document comes with a downloadable data package that contains all the data you need to run the examples presented. Instructions on where to get that package and how to set things up are given in *Downloading the tutorial datasets*.

Given the limited usefulness and general usage of this keyhole imaging mode, we provide here only one example, a point source dithered observation. We show how to reduce the sequence in two different ways:

- From the terminal, using the command line. (*Example 1-A*)
- From Python, using the DRAGONS classes and functions. (*Example 1-B*)

CHAPTER 2

Setting up and tutorial datasets

2.1 Downloading the tutorial datasets

All the data needed to run this tutorial are found in the tutorial's data package:

http://www.gemini.edu/sciops/data/software/datapkgs/gnirsimg_tutorial_datapkg-v1.tar

Download it and unpack it somewhere convenient.

```
cd <somewhere convenient>
tar xvf gnirsimg_tutorial_datapkg-v1.tar
bunzip2 gnirsimg_tutorial/playdata/*.bz2
```

The datasets are found in the subdirectory `gnirsimg_tutorial/playdata`, and we will work in the subdirectory named `gnirsimg_tutorial/playground`.

Note: All the raw data can also be downloaded from the Gemini Observatory Archive. Using the tutorial data package is probably more convenient.

2.2 Datasets descriptions

2.2.1 Dataset for example 1: Point source through the acquisition keyhole

This is a GNIRS acquisition keyhole imaging observation of a point source. The observation sequence uses dither-on-target. Dithered observations nearby in time will be used for sky subtraction of each frame.

The calibrations we use for this example include:

- Darks for the science frames
- Flats, as a sequence of lamps-on and lamps-off exposures

Here is the files breakdown. They are included in a tutorial data package. They can also be downloaded from the Gemini Observatory Archive (GOA).

Science	N20120117S0014-33 (J-band, on-target)
Science darks	N20120102S0538-547 (60 sec, like Science)
Flats	N20120117S0034-41 (lamps-on) N20120117S0042-49 (lamps-off)

A note about finding the darks in the GOA. Since GNIRS is not an imager and imaging through the keyhole is done only in extreme circumstances, the archive does not have calibration association rules for the darks in this case. One needs to manually search for the darks. Here is the search that was done to find the darks for this observation sequence:

- Set a date range around the dates of the science observations. In this case we used "20120101-20120131".
- Set **Instrument** to GNIRS.
- Set **Obs.Type** to DARK.
- Set the exposure time to 60 seconds.
- On the result table, select the darks with a "Pass" setting in the "QA" column.

CHAPTER 3

Example 1-A: Point source through keyhole - Using the “reduce” command

In this example we will reduce a GNIRS keyhole imaging observation of a point source using the “” command that is operated directly from the unix shell. Just open a terminal to get started.

This observation is a simple dither on target.

3.1 The dataset

If you have not already, download and unpack the tutorial’s data package. Refer to the *Downloading the tutorial datasets* for the links and simple instructions.

The dataset specific to this example is described in:

> *Dataset for example 1: Point source through the acquisition keyhole*.

Here is a copy of the table for quick reference.

Science	N20120117S0014-33 (J-band, on-target)
Science darks	N20120102S0538-547 (60 sec, like Science)
Flats	N20120117S0034-41 (lamps-on) N20120117S0042-49 (lamps-off)

3.2 Set up the Local Calibration Manager

DRAGONS comes with a local calibration manager and a local light weight database that uses the same calibration association rules as the Gemini Observatory Archive. This allows "" to make requests for matching **processed** calibrations when needed to reduce a dataset.

Let's set up the local calibration manager for this session.

In `~/.dragons/`, create or edit the configuration file `dragonsrc` as follow:

```
[calibs]
databases = <where_the_data_package_is>/gnirsimg_tutorial/playground/cal_manager.db␣
↪get
```

This simply tells the system where to put the calibration database, the database that will keep track of the processed calibrations we are going to send to it.

Note: ~ in the path above refers to your home directory. Also, don't miss the dot in `.dragons`.

Then initialize the calibration database:

```
caldb init
```

That's it. It is ready to use.

You can add processed calibrations with `caldb add <filename>` (we will later), list the database content with `caldb list`, and `caldb remove <filename>` to remove a file from the database (it will **not** remove the file on disk.) (See the "" documentation for more details.)

3.3 Create file lists

The first step is to create input file lists. The tool "" helps with that. It uses Astrodata tags and "" to select the files and send the filenames to a text file that can then be fed to "". (See the for information about Astrodata.)

First, navigate to the `playground` directory in the unpacked data package.

3.3.1 A list of the darks

There is only one set of 60-second darks in the data package. To create the list, one simply needs to select on the `DARK` tag:

```
dataselect ../playdata/*.fits --tags DARK -o darks60.lis
```

If there was a need to select specifically on the 60-second darks, the command would use the `exposure_time` descriptor:

```
dataselect ../playdata/*.fits --tags DARK --expr='exposure_time==60' -o darks60.lis
```

3.3.2 A list for the flats

The flats are a sequence of lamp-on and lamp-off exposures. We just send all of them to one list.

```
dataselect ../playdata/*.fits --tags FLAT -o flats.lis
```

3.3.3 A list for the science observations

The science frames are all the IMAGE non-FLAT frames in the data package. They are also the J filter images that are non-FLAT. And they are the ones with an object name GRB120116A. Those are all valid ways to select the science observations. Here we show all three ways as examples; of course, just one is required.

```
dataselect ../playdata/*.fits --tags IMAGE --xtags FLAT -o target.lis

dataselect ../playdata/*.fits --xtags FLAT --expr='filter_name=="J"' -o target.lis

dataselect ../playdata/*.fits --expr='object=="GRB120116A"' -o target.lis
```

Pick the one you prefer, they all yield the same list.

3.4 Master Dark

We first create the master dark for the science target, then add it to the calibration database. The name of the output master dark, N20120102S0538_dark.fits, is written to the screen at the end of the process.

```
reduce @darks60.lis
caldb add N20120102S0538_dark.fits
```

The @ character before the name of the input file is the "at-file" syntax. More details can be found in the documentation.

Note: The file name of the output processed dark is the file name of the first file in the list with *_dark* appended as a suffix. This is the general naming scheme used by "".

3.5 Master Flat Field

A GNIRS master flat is created from a series of lamp-on and lamp-off exposures. Each flavor is stacked, then the lamp-off stack is subtracted from the lamp-on stack.

We create the master flat field and add it to the calibration database as follows:

```
reduce @flats.lis
caldb add N20120117S0034_flat.fits
```

3.6 Science Observations

The science target is a point source. The sequence dithers on-target, moving the source across the thin keyhole aperture. The sky frames for each science image will be the adjacent dithered frames obtained within a certain time limit. The default for GNIRS keyhole images is "within 600 seconds". This can be seen by using "":

```
showpars ../playdata/N20120117S0014.fits associateSky
```

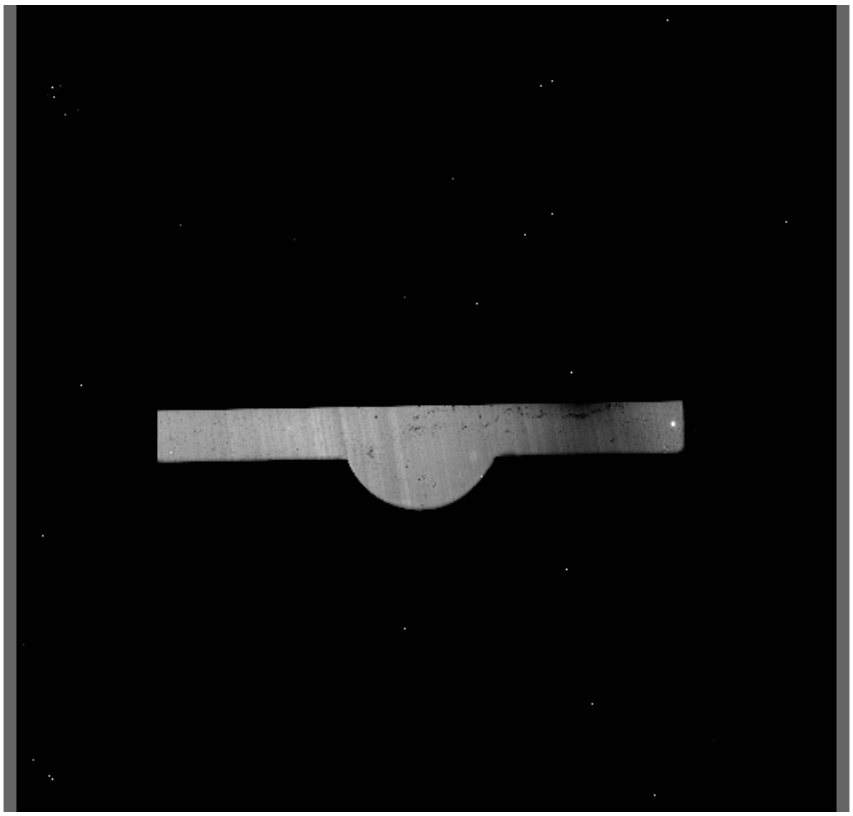
```
Dataset tagged as set(['RAW', 'GEMINI', 'NORTH', 'SIDEREAL', 'GNIRS', 'UNPREPARED', 'IMAGE'])
Settable parameters on 'associateSky':
========================================
 Name                   Current setting

suffix                '_skyAssociated'     Filename suffix
time                  600.0                Maximum association time (seconds)
        Valid Range = [0,inf)
distance              1.0                  Minimum association distance (arcsec)
        Valid Range = [0,inf)
min_skies             3                    Minimum number of sky frames to associate
        Valid Range = [0,inf)
max_skies             None                 Maximum number of sky frames to associate
        Valid Range = [1,inf)
use_all               False                Use all frames as skies?
```

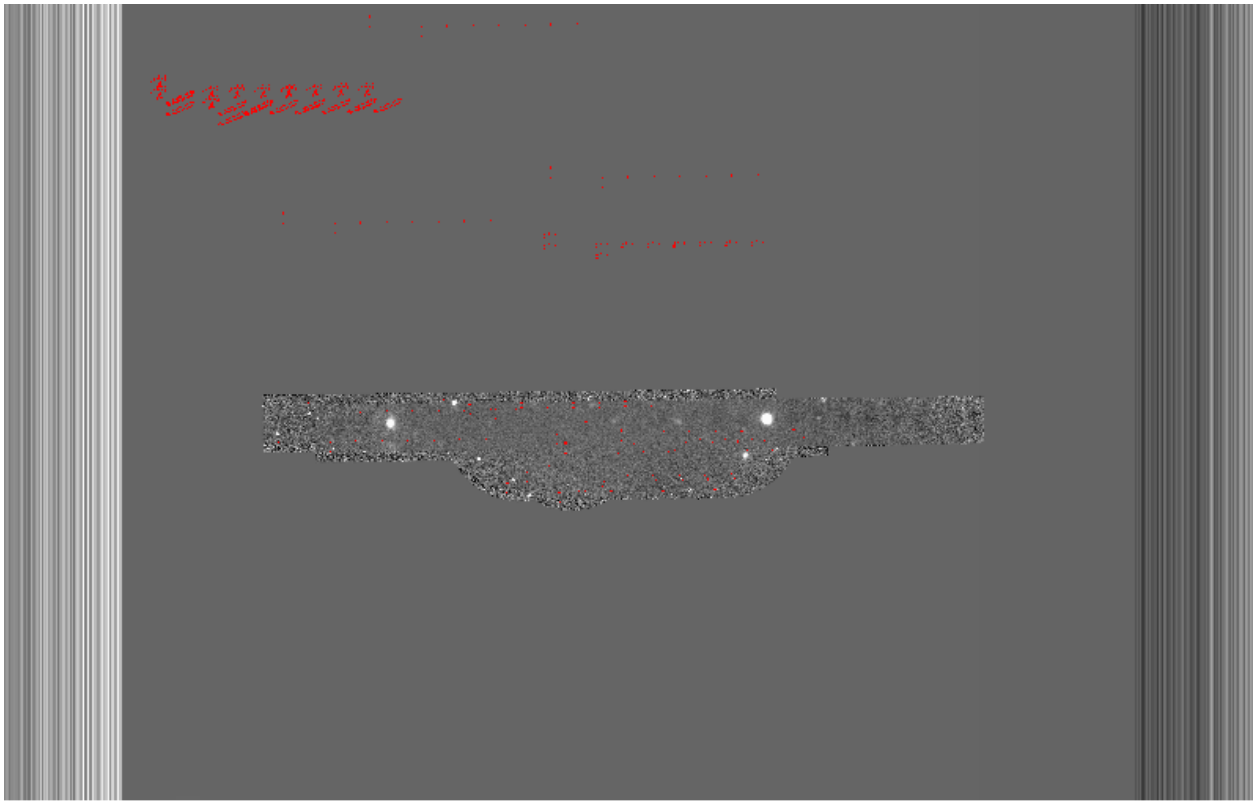
Both the master dark and the master flat are in our local calibration database. For any other Gemini facility instrument, they would both be retrieved automatically by the calibration manager. However, GNIRS not being an imager, and the keyhole being normally used only for acquisition, it turns out that there are no calibration association rules between GNIRS keyhole images and darks. This is a recently discovered limitation that we plan to fix in a future release. In the meantime, we are not stuck, we can simply specify the dark on the command line. The flat will be retrieved automatically.

```
reduce @target.lis --user_cal processed_dark:N20120102S0538_dark.fits
```

The output stack units are in electrons (header keyword BUNIT=electrons). The output stack is stored in a multi-extension FITS (MEF) file. The science signal is in the "SCI" extension, the variance is in the "VAR" extension, and the data quality plane (mask) is in the "DQ" extension.

Below are a raw image (top) and the final stacked image (bottom). The stack keeps all the pixels and is never cropped to only the common area. Of course the areas covered by less than the full stack of images will have a lower signal-to-noise.





CHAPTER 4

Example 1-B: Point source through keyhole - Using the “Reduce” class

A reduction can be initiated from the command line as shown in *Example 1-A: Point source through keyhole - Using the “reduce” command* and it can also be done programmatically as we will show here. The classes and modules of the RecipeSystem can be accessed directly for those who want to write Python programs to drive their reduction. In this example we replicate the command line reduction from Example 1-A, this time using the Python interface instead of the command line. Of course what is shown here could be packaged in modules for greater automation.

4.1 The dataset

If you have not already, download and unpack the tutorial’s data package. Refer to *Downloading the tutorial datasets* for the links and simple instructions.

The dataset specific to this example is described in:

> *Dataset for example 1: Point source through the acquisition keyhole*.

Here is a copy of the table for quick reference.

Science	N20120117S0014-33 (J-band, on-target)
Science darks	N20120102S0538-547 (60 sec, like Science)
Flats	N20120117S0034-41 (lamps-on) N20120117S0042-49 (lamps-off)

4.2 Setting up

First, navigate to the `playground` directory in the unpacked data package.

The first steps are to import libraries, set up the calibration manager, and set the logger.

4.2.1 Importing Libraries

```
import glob

# DRAGONS imports
from recipe_system.reduction.coreReduce import Reduce
from recipe_system import cal_service
from gempy.adlibrary import dataselect
```

The `dataselect` module will be used to create file lists for the darks, the flats and the science observations. The `cal_service` package is our interface to the calibration databases. Finally, the `Reduce` class is used to set up and run the data reduction.

4.2.2 Setting up the logger

We recommend using the DRAGONS logger. (See also *Double messaging issue*.)

```
from gempy.utils import logutils
logutils.config(file_name='gnirs_tutorial.log')
```

4.2.3 Set up the Local Calibration Manager

DRAGONS comes with a local calibration manager and a local light weight database that uses the same calibration association rules as the Gemini Observatory Archive. This allows the `Reduce` instance to make requests for matching **processed** calibrations when needed to reduce a dataset.

Let's set up the local calibration manager for this session.

In `~/.dragons/`, edit the configuration file `dragonsrc` as follow:

```
[calibs]
databases = <where_the_data_package_is>/gnirsimg_tutorial/playground/cal_manager.db␣
↪get
```

This tells the system where to put the calibration database, the database that will keep track of the processed calibration we are going to send to it.

Note: The tilde (~) in the path above refers to your home directory. Also, mind the dot in `.dragons`.

The calibration database is initialized and the calibration service is configured like this:

```
caldb = cal_service.set_local_database()
caldb.init()
```

The calibration service is now ready to use. If you need more details, check the "" documentation in the Recipe System User Manual.

4.3 Create file lists

The next step is to create input file lists. The tool `dataselect` helps with that. It uses Astrodata tags and descriptors to select the files and store the filenames to a Python list that can then be fed to the `Reduce` class. (See the for information about Astrodata and for a list of .)

The first list we create is a list of all the files in the `playdata` directory.

```
16 all_files = glob.glob('../playdata/*.fits')
17 all_files.sort()
```

We will search that list for files with specific characteristics. We use the `all_files list` as an input to the function `dataselect.select_data()` . The function's signature is:

```
select_data(inputs, tags=[], xtags=[], expression='True')
```

We show several usage examples below.

4.3.1 A list for the darks

There is only one set of 60-second darks in the data package. To create the list, one simply need to select on the `DARK` tag:

```
18 darks60 = dataselect.select_data(all_files, ['DARK'])
```

If there was a need to select specifically on the 60-second darks, the command would use the `exposure_time` descriptor:

```
19 darks60 = dataselect.select_data(
20     all_files,
21     ['DARK'],
22     [],
23     dataselect.expr_parser('exposure_time==60')
24 )
```

Note: All expressions need to be processed with `dataselect.expr_parser`.

4.3.2 A list for the flats

The flats are a sequence of lamp-on and lamp-off exposures. We just send all of them to one list.

```
25 flats = dataselect.select_data(all_files, ['FLAT'])
```

4.3.3 A list for the science observations

The science frames are all the `IMAGE` non-`FLAT` frames in the data package. They are also the `J` filter images that are non-`FLAT`. And they are the ones with an object name `GRB120116A`. Those are all valid ways to select the science observations. Here we show all three ways as examples; of course, just one is required.

```
target = dataselect.select_data(all_files, ['IMAGE'], ['FLAT'])

# Or...
target = dataselect.select_data(
    all_files,
    [],
    ['FLAT'],
    dataselect.expr_parser('filter_name=="J"')
)

# Or...
target = dataselect.select_data(
    all_files,
    [],
    [],
    dataselect.expr_parser('object=="GRB120116A"')
)
```

Pick the one you prefer, in this case, they all yield the same list.

4.4 Master Dark

We first create the master dark for the science target, then add it to the calibration database. The name of the output master dark is `N20120102S0538_dark.fits`. The output is written to disk and its name is stored in the `Reduce` instance. The calibration service expects the name of a file on disk.

```
reduce_darks = Reduce()
reduce_darks.files.extend(darks60)
reduce_darks.runr()

caldb.add_cal(reduce_darks.output_filenames[0])
```

The `Reduce` class is our reduction "controller". This is where we collect all the information necessary for the reduction. In this case, the only information necessary is the list of input files which we add to the `files` attribute. The `Reduce.runr{}` method is where the recipe search is triggered and where it is executed.

Note: The file name of the output processed dark is the file name of the first file in the list with *_dark* appended as a suffix. This the general naming scheme used by the `Recipe System`.

4.5 Master Flat Field

A GNIRS master flat is created from a series of lamp-on and lamp-off exposures. Each flavor is stacked, then the lamp-off stack is subtracted from the lamp-on stack.

We create the master flat field and add it to the calibration database as follows:

```
reduce_flats = Reduce()
reduce_flats.files.extend(flats)
reduce_flats.runr()

caldb.add_cal(reduce_flats.output_filenames[0])
```

4.6 Science Observations

The science target is a point source. The sequence dithers on-target, moving the source across the thin keyhole aperture. The sky frames for each science image will be the adjacent dithered frames obtained within a certain time limit. The default for GNIRS keyhole images is "within 600 seconds". This can be seen by using the "" command-line tool:

```
showpars ../playdata/N20120117S0014.fits associateSky
```

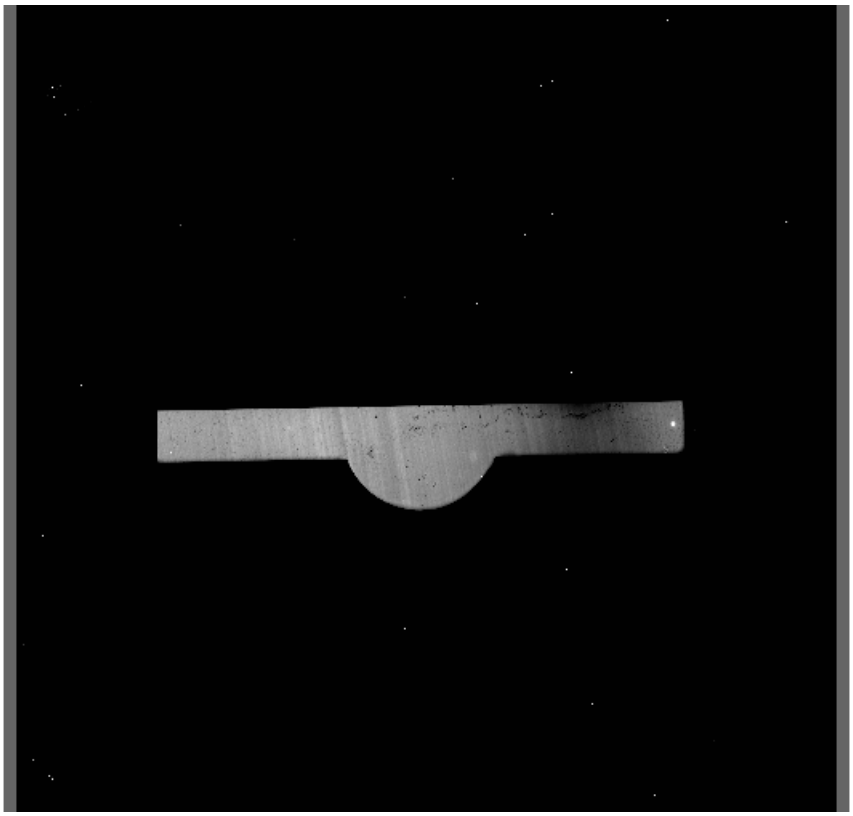
```
Dataset tagged as set(['RAW', 'GEMINI', 'NORTH', 'SIDEREAL', 'GNIRS', 'UNPREPARED', 'IMAGE'])
Settable parameters on 'associateSky':
=======================================
 Name                    Current setting

suffix                '_skyAssociated'     Filename suffix
time                  600.0                Maximum association time (seconds)
        Valid Range = [0,inf)
distance              1.0                  Minimum association distance (arcsec)
        Valid Range = [0,inf)
min_skies             3                    Minimum number of sky frames to associate
        Valid Range = [0,inf)
max_skies             None                 Maximum number of sky frames to associate
        Valid Range = [1,inf)
use_all               False                Use all frames as skies?
```

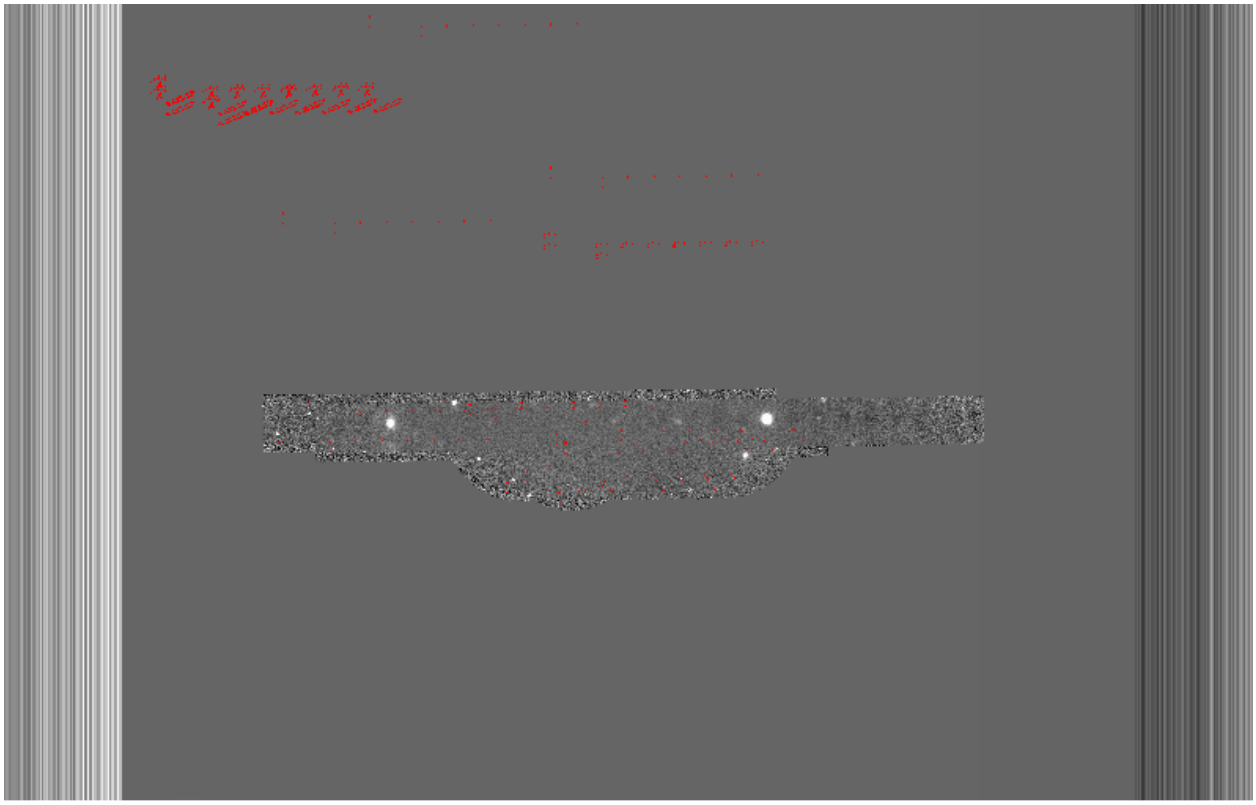
Both the master dark and the master flat are in our local calibration database. For any other Gemini facility instrument, they would both be retrieved automatically by the calibration manager. However, GNIRS not being an imager, and the keyhole being normally used only for acquisition, it turns out that there are no calibration association rules between GNIRS keyhole images and darks. This is a recently discovered limitation that we plan to fix in a future release. In the meantime, we are not stuck, we can simply specify the dark on the command line. The flat will be retrieved automatically.

```
from recipe_system.utils.reduce_utils import normalize_ucals
mycalibrations = ['processed_dark:N20120102S0538_dark.fits']

reduce_target = Reduce()
reduce_target.files.extend(target)
ucals_dict = normalize_ucals(reduce_target.files, mycalibrations)
reduce_target.ucals = ucals_dict
reduce_target.runr()
```

The output stack units are in electrons (header keyword BUNIT=electrons). The output stack is stored in a multi-extension FITS (MEF) file. The science signal is in the "SCI" extension, the variance is in the "VAR" extension, and the data quality plane (mask) is in the "DQ" extension.

Below are a raw image (top) and the final stacked image (bottom). The stack keeps all the pixels and is never cropped to only the common area. Of course the areas covered by less than the full stack of images will have a lower signal-to-noise.





CHAPTER 5

Tips and Tricks

This is a collection of tips and tricks that can be useful for reducing different data, or to do it slightly differently from what is presented in the example.

5.1 Bypassing automatic calibration association

We can think of two reasons why a user might want to bypass the calibration manager and the automatic processed calibration association. The first is to override the automatic selection, to force the use of a different processed calibration than what the system finds. The second is if there is a problem with the calibration manager and it is not working for some reason.

Whatever the specific situation, the following syntax can be used to bypass the calibration manager and set the input processed calibration yourself:

```
$ reduce @target.lis --user_cal processed_dark:N20120102S0538_dark.fits processed_
↪flat:N20120117S0034_flat.fits
```

The list of recognized processed calibration is:

- processed_arc
- processed_bias
- processed_dark
- processed_flat
- processed_fringe
- processed_standard

CHAPTER 6

Issues and Limitations

6.1 Missing Processed Dark Calibration Association Rules

GNIRS not being an imager, and the keyhole being normally used only for acquisition, it turns out that there are no calibration association rules between GNIRS keyhole images and darks. This is recently discovered limitation that we plan to fix in a future release. In the meantime, the user can simply specify the dark on the command line as shown in the previous chapter in the *Bypassing automatic calibration association* section.

6.2 Memory Issues

Some primitives use a lot of RAM memory and they can cause a crash. Memory management in Python is notoriously difficult. The DRAGONS's team is constantly trying to improve memory management within `astrodata` and the DRAGONS recipes and primitives. If an "Out of memory" crash happens to you, if possible for your observation sequence, try to run the pipeline on fewer images at the time, like for each dither pattern sequence separately.

Then to align and stack the pieces, run the `alignAndStack` recipe:

```
$ reduce @list_of_stacks -r alignAndStack
```

For GNIRS, this issue is very rare given that the GNIRS detector is fairly small and also rarely used for imaging, but it could happen when trying to reduce a very large number of frames in one go.

6.3 Double messaging issue

If you run `Reduce` without setting up a logger, you will notice that the output messages appear twice. To prevent this behaviour set up a logger. This will send one of the output stream to a file, keeping the other on the screen. We recommend using the DRAGONS logger located in the `logutils` module and its `config()` function:

```
1 from gempy.utils import logutils
2 logutils.config(file_name='gnirs_tutorial.log')
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search