DRAGONS Tutorial - Flamingos-2 Data Reduction

Release 3.0.0

Bruno Quint

October 2021

Table of Contents

Document ID

PIPE-USER-115_F2Img-DRTutorial

This is a brief tutorial on how to reduce Flamingos-2 images using DRAGONS (Data Reduction for Astronomy from Gemini Observatory North and South). It is based on information found in the GEMINI Flamingos-2 WebPage and in the DRAGONS Documentation on Read The Docs.

CHAPTER 1

Introduction

This tutorial covers the basics of reducing Flamingos-2 data using DRAGONS.

The next two sections explain what are the required software and the data set that we use throughout the tutorial. *Chapter 2: Data Reduction* contains a quick example on how to reduce data using the DRAGONS command line tools. *Chapter 3: Reduction with API* shows how we can reduce the data using DRAGONS packages from within Python.

1.1 Software Requirements

Before you start, make sure you have DRAGONS properly installed and configured on your machine. You can test that by typing the following commands:

```
$ conda activate dragons
$ python -c "import astrodata"
```

Where `dragons` is the name of the conda environment where DRAGONS has been installed. If you have an error message, make sure:

- Conda is properly installed;
- A Conda Virtual Environment is properly created and is active;
- AstroConda (STScI) is properly installed within the Virtual Environment;
- DRAGONS was successfully installed within the Conda Virtual Environment;

1.2 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/f2img_tutorial_datapkg-v1.tar

Download it and unpack it somewhere convenient.

```
cd <somewhere convenient>
tar xvf f2img_tutorial_datapkg-v1.tar
bunzip2 f2img_tutorial/playdata/*.bz2
```

The datasets are found in the subdirectory `f2img_tutorial/playdata`, and we will work in the subdirectory named `f2img_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 but if you really want to learn how to search for and retrieve the data yourself, see the step-by-step instructions in the appendix, *Downloading from the Gemini Observatory Archive*.

1.3 About the dataset

1.3.1 Dither-on-target

This is a Flamingos-2 imaging observation of a star and distant galaxy field with dither on target for sky subtraction.

The calibrations we use in this example include:

- Darks for the science frames.
- Flats, as a sequence of lamps-on and lamps-off exposures.
- Short darks to use with the flats to create a bad pixel mask.

The table below contains a summary of the files needed for this example:

Science	S20131121S0075-083	Y-band, 120 s
Darks	S20131121S0369-375	2 s, short darks for BPM
	S20131120S0115-120 S20131121S0010 S20131122S0012 S20131122S0438-439	120 s, for science data
Flats	S20131129S0320-323	20 s, Lamp On, Y-band
	S20131126S1111-116	20 s, Lamp Off, Y-band

CHAPTER 2

Data Reduction with “reduce”

This chapter will guide you on reducing **Flamingos-2 imaging data** using command line tools. In this example we reduce a Flamingos-2 observation of a star and distant galaxy field. The observation is a simple dither-on-target sequence. Just open a terminal to get started.

While the example cannot possibly cover all situations, it will help you get acquainted with the reduction of Flamingos-2 data with DRAGONS. We encourage you to look at the *Tips and Tricks* and *Issues and Limitations* chapters to learn more about F2 data reduction.

DRAGONS installation comes with a set of useful scripts that are used to reduce astronomical data. The most important script is called , which is extensively explained in the . It is through that command that a DRAGONS reduction is launched.

For this tutorial, we will be also using other support tools like:

-
-
-
-

2.1 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:

About the dataset

Here is a copy of the table for quick reference.

Science	S20131121S0075-083	Y-band, 120 s
Darks	S20131121S0369-375	2 s, short darks for BPM
	S20131120S0115-120 S20131121S0010 S20131122S0012 S20131122S0438-439	120 s, for science data
Flats	S20131129S0320-323	20 s, Lamp On, Y-band
	S20131126S1111-116	20 s, Lamp Off, Y-band

2.2 Set up the Local Calibration Manager

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

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

In `~/.geminidr/`, create or edit the configuration file `rsys.cfg` as follow:

```
[calibs]
standalone = True
database_dir = ${path_to_my_data}/f2img_tutorial/playground
```

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

Then initialize the calibration database:

```
caldb init
```

That's it! It is ready to use! You can check the configuration and confirm the setting with `caldb config`.

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). For more the details, check the Recipe System Local Calibration Manager documentation.

2.3 Check files

For this example, all the raw files we need are in the same directory called `../playdata/`. Let us learn a bit about the data we have.

Ensure that you are in the `playground` directory and that the `conda` environment that includes DRAGONS has been activated.

Let us call the command tool :

```
$ typewalk -d ../playdata/

directory:  /path_to_my_files/f2img_tutorial/playdata
     S20131120S0115.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (F2)␣
↪(GEMINI) (NON_SIDEREAL) (RAW) (SOUTH) (UNPREPARED)
     ...
     S20131121S0075.fits ............... (F2) (GEMINI) (IMAGE) (RAW) (SIDEREAL)␣
↪(SOUTH) (UNPREPARED)
     ...
     S20131121S0369.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (F2)␣
↪(GEMINI) (NON_SIDEREAL) (RAW) (SOUTH) (UNPREPARED)
     ...
     S20131126S1111.fits ............... (AZEL_TARGET) (CAL) (F2) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NON_SIDEREAL) (RAW) (SOUTH) (UNPREPARED)
     ...
     S20131129S0320.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (F2) (FLAT)␣
↪(GCALFLAT) (GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NON_SIDEREAL) (RAW) (SOUTH)␣
↪(UNPREPARED)
     ...
Done DataSpider.typewalk(..)
```

This command will open every FITS file within the directory passed after the `-d` flag (recursively) and will print an unsorted table with the file names and the associated tags. For example, calibration files will always have the `CAL` tag. Flat images will always have the `FLAT` tag. Dark files will have the `DARK` tag. This means that we can start getting to know a bit more about our data set just by looking at the tags. The output above was trimmed for presentation.

2.4 Create file lists

This data set contains science and calibration frames. For some programs, it could have different observed targets and different exposure times depending on how you like to organize your raw data.

The DRAGONS data reduction pipeline does not organize the data for you. You have to do it. DRAGONS provides tools to help you with that.

The first step is to create lists that will be used in the data reduction process. For that, we use . Please, refer to the documentation for details regarding its usage.

2.4.1 Two lists for the darks

Our data set contains two sets of DARK files: some 120-seconds darks matching the science data and some 2-second darks to create the bad pixel mask (BPM). If you did not know the exposure times of the darks, you could send the results to the command line tool as follow to get the information:

```
$ dataselect --tags DARK ../playdata/*.fits | showd -d exposure_time
-----------------------------------------------
filename                          exposure_time
-----------------------------------------------
../playdata/S20131120S0115.fits           120.0
../playdata/S20131120S0116.fits           120.0
../playdata/S20131120S0117.fits           120.0
...
../playdata/S20131121S0369.fits             2.0
../playdata/S20131121S0370.fits             2.0
../playdata/S20131121S0371.fits             2.0
...
../playdata/S20131122S0012.fits           120.0
../playdata/S20131122S0438.fits           120.0
../playdata/S20131122S0439.fits           120.0
```

(The list has been shorten for presentation.)

The | is the Unix "pipe" operator and it is used to pass output from to .

Let us go ahead and create our two list of darks. The following line creates a list of dark files that have exposure time of 120 seconds:

```
$ dataselect --tags DARK --expr "exposure_time==120" ../playdata/*.fits -o darks_120s.
↪list
```

`--expr` is used to filter the files based on their . Here we are selecting files with exposure time of 120 seconds. You can repeat the same command with the other exposure time to get the list of short darks.

```
$ dataselect --tags DARK --expr "exposure_time==2" ../playdata/*.fits -o darks_002s.
↪list
```

2.4.2 A list for the flats

Now let us create the list containing the flat files:

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

We know that our dataset has only one filter (Y-band). If our dataset contained data with more filters, we would have had to use the `--expr` option to select the appropriate filter as follow:

```
$ dataselect --tags FLAT --expr "filter_name=='Y'" ../playdata/*.fits -o flats_Y.list
```

Note: Flamingos-2 Y, J and H flat fields are created from lamps-on and lamps-off flats. The software will sort them out, so put all lamps-on, lamp-off flats, in the list and let the software use them appropriately.

2.4.3 A list for the science observations

Finally, we want to create a list of the science targets. We are looking for files that are not calibration frames. To exclude them from our selection we can use the `--xtags`, e.g., `--xtags CAL`.

```
$ dataselect --xtags CAL ../playdata/*.fits -o sci_images.list
```

Remember that you can use the `--expr` option to select targets with different names (`object`) or exposure times (`exposure_time`), or use it with any of the datasets .

2.5 Create a Master Dark

We start the data reduction by creating a master dark for the science data. Here is how you reduce the 120 s dark data into a master dark:

```
$ reduce @darks_120s.list
```

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

The master dark is added to the local calibration manager using the following command:

```
$ caldb add S20131120S0115_dark.fits
```

Now will be able to find this processed dark when needed to process other observations.

Note: The master dark will be saved in the same folder where was called *and* inside the `./calibrations/processed_dark` folder. The latter location is to cache a copy of the file. This applies to all the processed calibration.

Some people might prefer adding the copy in the *calibrations* directory as it is safe from a *rm* *, for example.

```
$ caldb add ./calibrations/processed_dark/S20131120S0115_dark.fits
```

2.6 Create a Bad Pixel Mask

The Bad Pixel Mask (BPM) can be built using a set of flat images with the lamps on and off and a set of short exposure dark files. Here, our shortest dark files have 2 second exposure time. Again, we use the command to produce the BPMs.

It is important to note that the recipe library association is done based on the nature of the **first file in the input list**. Since the recipe to make the BPM is located in the recipe library for flats, the first item in the list must be a flat.

For Flamingos-2, the filter wheel's location is such that the choice of filter does not interfere with the results. Here we have Y-band flats, so we will use Y-band flats.

```
$ reduce @flats_Y.list @darks_002s.list -r makeProcessedBPM
```

The `-r` tells which recipe from the recipe library for F2-FLAT to use. If not specified the system will use the default recipe which is the one that produces a master flat, this is not what we want here. The output image will be saved in the current working directory with a `_bpm` suffix.

The local calibration manager does not yet support BPMs so we cannot add it to the database. It is a future feature. Until then we have to pass it manually to `reduce` to use it, as we will show below.

2.7 Create a Master Flat Field

The F2 Y-band master flat is created from a series of lamp-on and lamp-off exposures. They should all have the same exposure time. Each flavor is stacked (averaged), then the lamp-off stack is subtracted from the lamp-on stack and the

result normalized.

We create the master flat field and add it to the calibration manager as follow:

```
$ reduce @flats_Y.list -p addDQ:user_bpm="S20131129S0320_bpm.fits"
$ caldb add S20131129S0320_flat.fits
```

Here, the `-p` flag tells to set the input parameter `user_bpm` of the `addDQ` primitive to the filename of the BPM we have just created. There will be a message "WARNING - No static BPMs defined". This is normal. This is because F2 does not have a static BPM that is distributed with the package. Your user BPM is the only one that is available.

2.8 Reduce the Science Images

Now that we have the master dark and the master flat, we can tell to process our science data. will look at the local database for calibration files.

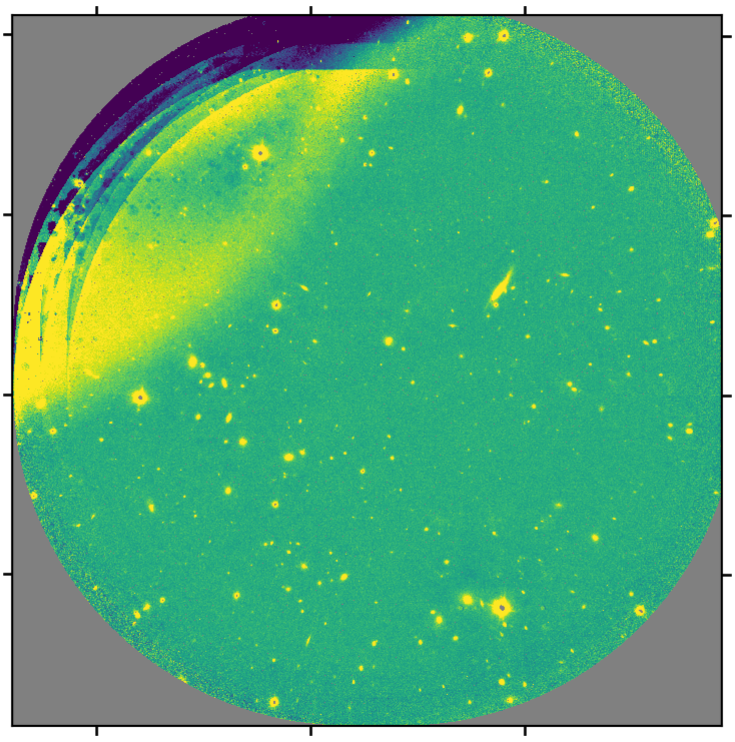
```
$ reduce @sci_images.list -p addDQ:user_bpm="S20131129S0320_bpm.fits"
```

This command retrieves the master dark and the master flat, and applies them to the science data. For sky subtraction, the software analyses the sequence to establish whether this is a dither-on-target or an offset-to-sky sequence and then proceeds accordingly. Finally, the sky-subtracted frames are aligned and stacked together. Sources in the frames are used for the alignment.

The final product file will have a `_stack.fits` suffix and it is shown below.

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.

Warning: The upper-left quadrant of this science sequence is rather messy. This is caused by the PWFS2 guide probe (see *Emission from PWFS2 guide probe*). Photometry in this portion of the image is likely to be seriously compromised.



CHAPTER 3

Reduction using API

There may be cases where you would be interested in accessing the DRAGONS Application Program Interface (API) directly instead of using the command line wrappers to reduce your data. Here we show you how to do the same reduction we did in the previous chapter but using the API.

3.1 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:

> *About the dataset*.

Here is a copy of the table for quick reference.

Science	S20131121S0075-083	Y-band, 120 s
Darks	S20131121S0369-375	2 s, short darks for BPM
	S20131120S0115-120 S20131121S0010 S20131122S0012 S20131122S0438-439	120 s, for science data
Flats	S20131129S0320-323	20 s, Lamp On, Y-band
	S20131126S1111-116	20 s, Lamp Off, Y-band

3.2 Setting Up

3.2.1 Importing Libraries

We first import the necessary modules and classes:

```
1 import glob
2
3 from gempy.adlibrary import dataselect
4 from recipe_system import cal_service
5 from recipe_system.reduction.coreReduce import Reduce
```

Importing `print_function` is for compatibility with the Python 2.7 print statement. If you are working with Python 3, it is not needed, but importing it will not break anything.

`glob` is Python built-in packages. It will be used to return a `list` with the input file names.

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

3.2.2 Setting up the logger

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

```
8 from gempy.utils import logutils
9 logutils.config(file_name='f2_data_reduction.log')
```

3.2.3 Setting up the Calibration Service

Before we continue, let's be sure we have properly setup our calibration database and the calibration association service.

First, check that you have already a `rsys.cfg` file inside the `~/.geminidr/`. It should contain:

```
[calibs]
standalone = True
database_dir = ${path_to_my_data}/f2img_tutorial/playground
```

This tells the system where to put the calibration database. This database will keep track of the processed calibrations as we add them to it.

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

The calibration database is initialized and the calibration service is configured as follow:

```
caldb = cal_service.CalibrationService()
caldb.config()
caldb.init()

cal_service.set_calservice()
```

The calibration service is now ready to use. If you need more details, check the section on using the API in the .

3.3 Create list of files

Next step is to create lists of files that will be used as input to each of the data reduction steps. Let us start by creating a `list` of all the FITS files in the directory `../playdata/`.

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

The `sort()` method simply re-organize the list with the file names and is an optional step. Before you carry on, you might want to do `print(all_files)` to check if they were properly read.

Now we can use the `all_files list` as an input to `select_data()`. The `dataselect.select_data()` function signature is:

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

3.3.1 Two list for the darks

We select the files that will be used to create a master dark for the science observations, those with an exposure time of 120 seconds.

```
dark_files_120s = dataselect.select_data(
    all_files,
    ['F2', 'DARK', 'RAW'],
    [],
    dataselect.expr_parser('exposure_time==120')
)
```

Above we are requesting data with tags `F2`, `DARK`, and `RAW`, though since we only have F2 raw data in the directory, `DARK` would be sufficient in this case. We are not excluding any tags, as represented by the empty list `[]`. The expression setting the exposure time criterion needs to be processed through the `dataselect` expression parser, `expr_parser()`.

We repeat the same syntax for the 2-second darks:

```
dark_files_2s = dataselect.select_data(
    all_files,
    ['F2', 'DARK', 'RAW'],
    [],
    dataselect.expr_parser('exposure_time==2')
)
```

3.3.2 A list for the flats

Now you must create a list of FLAT images for each filter. The expression specifying the filter name is needed only if you have data from multiple filters. It is not really needed in this case.

```
list_of_flats_Y = dataselect.select_data(
     all_files,
     ['FLAT'],
     [],
     dataselect.expr_parser('filter_name=="Y"')
)
```

3.3.3 A list for the science data

Finally, the science data can be selected using:

```
list_of_science_images = dataselect.select_data(
    all_files,
    ['F2'],
    [],
    dataselect.expr_parser('(observation_class=="science" and filter_name=="Y")')
)
```

The filter name is not really needed in this case since there are only Y-band frames, but it shows how you could have two selection criteria in the expression.

3.4 Create a 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 `N20160102S0423_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(dark_files_120s)
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 `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 is the general naming scheme used by the `Recipe System`.

3.5 Create a Bad Pixel Mask

By default, for F2 imaging data, an illumination mask will be added to the data quality plane to identify the pixels beyond the circular aperture as "non-illuminated". The package does not have a default bad pixel mask for F2 but the user can easily create a fresh bad pixel mask from the flats and recent short darks.

The Bad Pixel Mask is created using as follow:

```
46 reduce_bpm = Reduce()
47 reduce_bpm.files.extend(list_of_flats_Y)
48 reduce_bpm.files.extend(dark_files_2s)
49 reduce_bpm.recipename = 'makeProcessedBPM'
50 reduce_bpm.runr()
51
52 bpm_filename = reduce_bpm.output_filenames[0]
```

The flats must be passed first to the input list to ensure that the recipe library associated with F2 flats is selected. We are setting the recipe name to `makeProcessedBPM` to select that recipe from the recipe library instead of the using the default (which would create a master flat).

The BPM produced is named `S20131129S0320_bpm.fits`.

The local calibration manager does not yet support BPMs so we cannot add it to the database. It is a future feature. Until then we have to pass it manually to the `Reduce` instance to use it, as we will show below.

3.6 Create a Master Flat Field

A F2 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 and the result normalized.

We create the master flat field and add it to the calibration manager as follow:

```
53 reduce_flats = Reduce()
54 reduce_flats.files.extend(list_of_flats_Y)
55 reduce_flats.uparms = [('addDQ:user_bpm', bpm_filename)]
56 reduce_flats.runr()
57
58 caldb.add_cal(reduce_flats.output_filenames[0])
```

Note how we pass in the BPM we created in the previous step. The `addDQ` primitive, one of the primitives in the recipe, has an input parameter named `user_bpm`. We assign our BPM to that input parameter. The value of `uparms` needs to be a `list` of `Tuples`.

Once `runr()` is finished, we add the master flat to the calibration manager (line 59).

3.7 Reduce the Science Images

The science observation uses a dither-on-target pattern. The sky frames will be derived automatically for each science frame from the dithered frames.

The master dark and the master flat will be retrieved automatically from the local calibration database. Again, the user BPM needs to be specified as the `user_bpm` argument to `addDQ`.

We use similar commands as before to initiate a new reduction to reduce the science data:

```
reduce_target = Reduce()
reduce_target.files.extend(list_of_science_images)
reduce_target.uparms = [('addDQ:user_bpm', bpm_filename)]
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.

CHAPTER 4

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.

4.1 Flatfields

4.1.1 Y, J, and H-bands

Flamingos-2 Y, J and H master flats are created from lamps-on and lamps-off flats. Both types are passed in together to the "" command. The order does not matter. The software separates the lamps-on and lamps-off flats and use them appropriately.

4.1.2 K-band

For K-band master flats, lamp-off flats and darks are used. In that case both flats (lamp-off only for K-band) and darks need to be fed to "". The darks' exposure time must match that of the flats. The first input file to "" must be a flat for the correct recipe library to be selected. After that the software will sort out how to use the inputs appropriately to produce the flat. For example:

```
$ reduce @flats_K.list @darks_for_flats.list
```

The K-band thermal emission from the GCAL shutter depends upon the temperature at the time of the exposure, and includes some spatial structure. Therefore the distribution of emission is not necessarily consistent, except for sequential exposures. So it is best to combine lamp-off exposures from a single day.

4.2 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 @sci_images.list --user_cal processed_dark:S20131120S0115_dark.fits␣
↪processed_flat:S20131129S0320_flat.fits
```

The list of recognized processed calibration is:

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

CHAPTER 5

Issues and Limitations

5.1 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
```

5.2 Emission from PWFS2 guide probe

The PWFS2 guide probe leaves a signature on imaging data that cannot be removed. Ideally, one would be using the OIWFS, the On-Instrument Wave Front Sensor, but there has been issues with it. See https://www.gemini.edu/sciops/instruments/flamingos2/status-and-availability for the current status of the instrument. The effect of the PWFS2 guide probe will in many cases compromise photometry in the region affected.

5.3 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 `gempy.utils.logutils` module and its `config()` function:

```
from gempy.utils import logutils
logutils.config(file_name='f2_data_reduction.log')
```

CHAPTER 6

Downloading from the Gemini Observatory Archive

For this tutorial we provide a pre-made package with all the necessary data. Here we show how one can search and download the data directly from the archive, like one would have to do for their own program.

If you are just interested in trying out the tutorial, we recommend that you download the pre-made package (*Downloading the tutorial datasets*) instead of getting everything manually.

6.1 Query and Download

This tutorial uses 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; daytime calibrations, darks and GCAL flats, were obtained as well. Leggett, et al. (2015) briefly describes the data reduction procedures they followed, which are similar to those described in this tutorial.

The first step of any reduction is to retrieve the data from the Gemini Observatory Archive (GOA). For more details on using the Archive, check its Help Page.

6.1.1 Science data

Navigate to the GOA webpage search form. Put the data label **GS-2013B-Q-15-39** in the `PROGRAM ID` text field, and press the `Search` button in the middle of the page. The page will refresh and display a table with all the data for this dataset. Since the amount of data is unnecessarily large for a tutorial (162 files, 0.95 Gb), we have narrowed our search to only the Y-band data by setting the `Instrument` drop-down menu to **F2** and the `Filter` drop-down menu to **Y**. Now we have only 9 files, 0.05 Gb.

You can also copy the URL below and paste it on browser to see the search results:

```
https://archive.gemini.edu/searchform/GS-2013B-Q-15-39/RAW/cols=CTOWEQ/filter=Y/
↪notengineering/F2/NotFail
```

At the bottom of the page, you will find a button saying *Download all 9 files totalling 0.05 Gb*. Click on it to download a *.tar* file with all the data.

6.1.2 Calibrations

Matching calibration files can be obtained by clicking on the *Load Associated Calibrations* tab. For this data, we need the 120-second darks (for 120-second science data). We also need the Y-band flats; the series there is collection of lamp-on and lamp-off flats.

Select the darks and the Y-band flats at the top of the returned list by checking the little boxes on the left. Scroll down and click "Download Marked Files"

Finally, you will need a set of short dark frames in order to create the Bad Pixel Masks (BPM). For that, we will have to perform a search ourselves in the archive.

First remove the Program ID. The science data was obtained on November 21, 2013. So, we set the "UTC Date" to a range of a few days around the observations date. This and other settings are:

- Program ID: <empty>
- UTC Date: 20131120-20131122
- Instrument: F2
- Obs. Type: Dark
- Filter: Any

Hit the "Search" button. You can sort the list by exposure time by clicking on header of the "ExpT" column. Several 2-second darks show up. Some were even taken on the same date as the science data (20131121). Select those, and download them as we did before for the other calibrations.

6.2 Unpacking the data

Now, copy all the `.tar` files to the same place in your computer. Then use `tar` and `bunzip2` commands to decompress them. For example:

```
$ cd ${path_to_my_data}/
$ tar -xf gemini_data.tar
$ bunzip2 *.fits.bz2
```

(The tar files names may differ slightly depending on how you selected and downloaded the data from the Gemini Archive.)

Note: If you are using the manually selected data to run the tutorial, please remember to put all the data in a directory called `playdata`, and create a parallel directory of running the tutorial called `playground`. The tutorial makes assumption as to where everything is located.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search