DRAGONS Tutorial - GMOS Data Reduction

Release 3.0.4

Bruno Quint, Kathleen Labrie

November 2022

Table of Contents

Document ID

PIPE-USER-116_GMOSImg-DRTutorial

This is a brief tutorial on how to reduce GMOS images using DRAGONS. It is based on information found in the GEMINI GMOS WebPage and in the DRAGONS Documentation on Read The Docs.

CHAPTER 1

Introduction

This tutorial covers the basics of reducing GMOS (Gemini Multi-Object Spectrographs) data using .

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 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 should be 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/gmosimg_tutorial_datapkg-v1.tar

Download it and unpack it somewhere convenient.

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

The datasets are found in the subdirectory `gmosimg_tutorial/playdata`, and we will work in the subdirectory named `gmosimg_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

The data used for this tutorial is a dithered sequence on a starry field.

The table below contains a summary of the dataset downloaded in the previous section:

Science	N20170614S0201-205	10 s, i-band
Bias	N20170613S0180-184 N20170615S0534-538	
Twilight Flats	N20170702S0178-182	40 to 16 s, i-band

CHAPTER 2

Data Reduction

This chapter will guide you on reducing **GMOS imaging data** using command line tools. In this example we reduce a GMOS observation star 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 GMOS data with DRAGONS. We encourage you to look at the *Tips and Tricks* and *Issues and Limitations* chapters to learn more about GMOS data reduction.

DRAGONS installation comes with a set of 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 , 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	N20170614S0201-205	10 s, i-band
Bias	N20170613S0180-184 N20170615S0534-538	
Twilight Flats	N20170702S0178-182	40 to 16 s, i-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/gmosimg_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 **only** from the database (it will **not** remove the file on disk). For more the details, check the "" documentation in the Recipe System: User's Manual.

Note: If you have problems setting up "" or want to bypass it for another reason, you can check the *Bypassing automatic calibration association* section.

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:  /data/workspace/gmosimg_tutorial/playdata
 N20170613S0180.fits ............... (AT_ZENITH) (AZEL_TARGET) (BIAS) (CAL) (GEMINI)␣
↪(GMOS) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 ...
 N20170614S0201.fits ............... (GEMINI) (GMOS) (IMAGE) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 ...
 N20170615S0534.fits ............... (AT_ZENITH) (AZEL_TARGET) (BIAS) (CAL) (GEMINI)␣
↪(GMOS) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 ...
 N20170702S0182.fits ............... (CAL) (FLAT) (GEMINI) (GMOS) (IMAGE) (NORTH)␣
↪(RAW) (SIDEREAL) (TWILIGHT) (UNPREPARED)
Done DataSpider.typewalk(..)
```

This command will open every FITS file within the folder 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. This means that we can start getting to know a bit more about our data set just by looking 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 List of Biases

The bias files are selected with "":

```
$ dataselect --tags BIAS ../playdata/*.fits -o list_of_bias.txt
```

2.4.2 List of Flats

Now we can do the same with the FLAT files:

```
$ dataselect --tags FLAT ../playdata/*.fits -o list_of_flats.txt
```

If your dataset has flats obtained with more than one filter, you can add the `--expr 'filter_name=="i"'` expression to get only the flats obtained within the i-band. For example:

```
$ dataselect --tags FLAT --expr 'filter_name=="i"' ../playdata/*.fits -o list_of_
↪flats.txt
```

2.4.3 List for science data

The rest is the data with your science target. The simplest way, in this case, of creating a list of science frames is excluding everything that is a calibration:

```
$ dataselect --xtags CAL ../playdata/*.fits -o list_of_science.txt
```

This will work for our dataset because we know that a single target was observed with a single filter and with the same exposure time. But what if we don't know that?

We can check it by passing the "" output to the "" command line using a "pipe" (`|`):

```
$ dataselect --expr 'observation_class=="science"' ../playdata/*.fits | showd -d␣
↪object,exposure_time
------------------------------------------------------------
filename                                 object   exposure_time
------------------------------------------------------------
../playdata/N20170614S0201.fits   starfield            10.0
../playdata/N20170614S0202.fits   starfield            10.0
../playdata/N20170614S0203.fits   starfield            10.0
../playdata/N20170614S0204.fits   starfield            10.0
../playdata/N20170614S0205.fits   starfield            10.0
```

The `-d` flag tells "" which "" will be printed for each input file. As you can see, we have only observed target and only exposure time.

To select on target name and exposure time, specify the criteria in the `expr` field of "":

```
$ dataselect --expr '(object=="starfield" and exposure_time==10.)' ../playdata/*.fits␣
↪-o list_of_science.txt
```

We have our input lists and we have initialized the calibration database, we are ready to reduce the data.

Please make sure that you are still in the `playground` directory.

2.5 Create a Master Bias

We start the data reduction by creating a master bias for the science data. It can be created and added to the calibration database using the commands below:

```
$ reduce @list_of_bias.txt
$ caldb add N20170613S0180_bias.fits
```

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

To check that the master bias was added to the database, use `caldb list`.

Note: The master bias will be saved in the same folder where "" was called *and* inside the `./calibrations/processed_bias` 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_bias/N20170613S0180_bias.fits
```

Note: "" uses the first filename in the input list as basename and adds `_bias` as a suffix to it. So if your first filename is, for example, `N20170613S0180.fits`, the output will be *N20170613S0180_bias.fits*'.

2.6 Create a Master Flat Field

Twilight flats images are used to produce an imaging master flat and the result is added to the calibration database.

```
$ reduce @list_of_flats.txt
$ caldb add N20170702S0178_flat.fits
```

Note "" will query the local calibration manager for the master bias and use it in the data reduction.

Once finished you will have the master flat in the current work directory and inside `./calibrations/processed_flat`. It will have a `_flat` suffix.

2.7 Create Master Fringe Frame

Warning: The dataset used in this tutorial does not require fringe correction so we skip this step. To find out how to produce a master fringe frame, see *Create Master Fringe Frame* in the *Tips and Tricks* chapter.

2.8 Reduce Science Images

Once we have our calibration files processed and added to the database, we can run `reduce` on our science data:

```
$ reduce @list_of_science.txt
```

This command will generate bias and flat corrected files and will stack them. If a fringe frames is needed this command will apply the correction. The stacked image will have the `_stack` suffix.

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.

Note: Depending on your version of Astropy, you might see a lot of Astropy warnings about headers and coordinates system. You can safely ignore them.

Below are one of the raw images and the final stack:

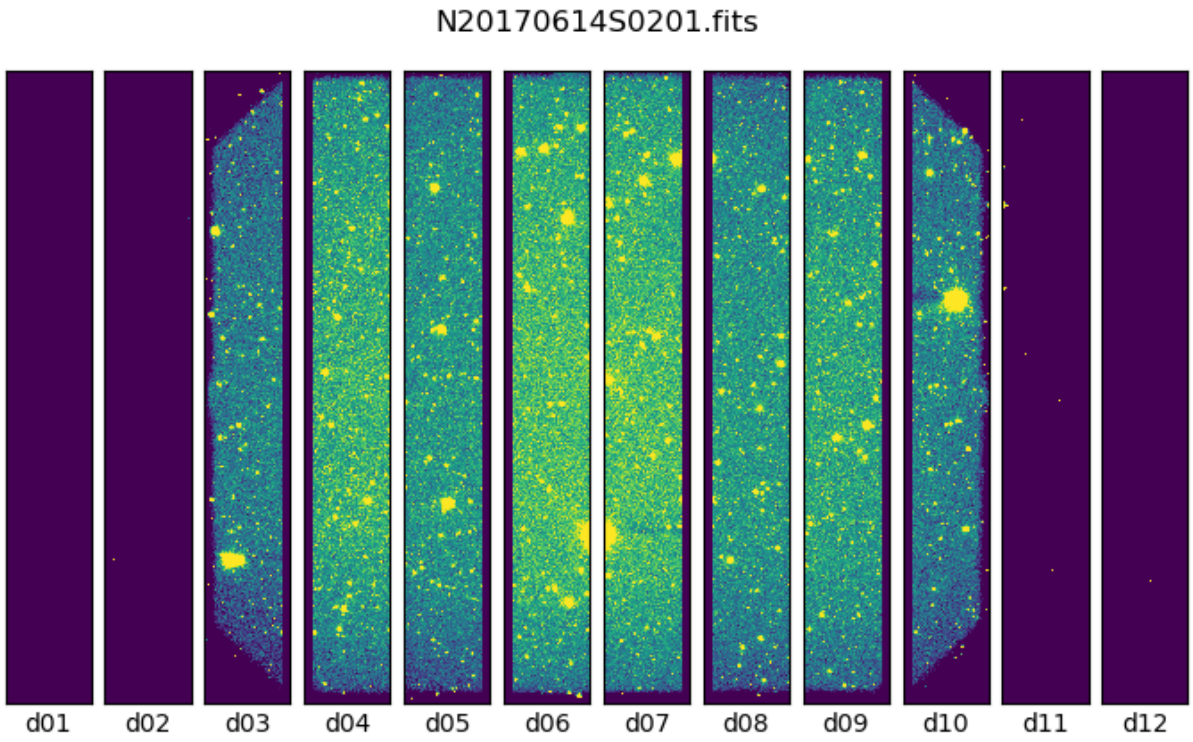

Fig. 1: One of the multi-extensions files.

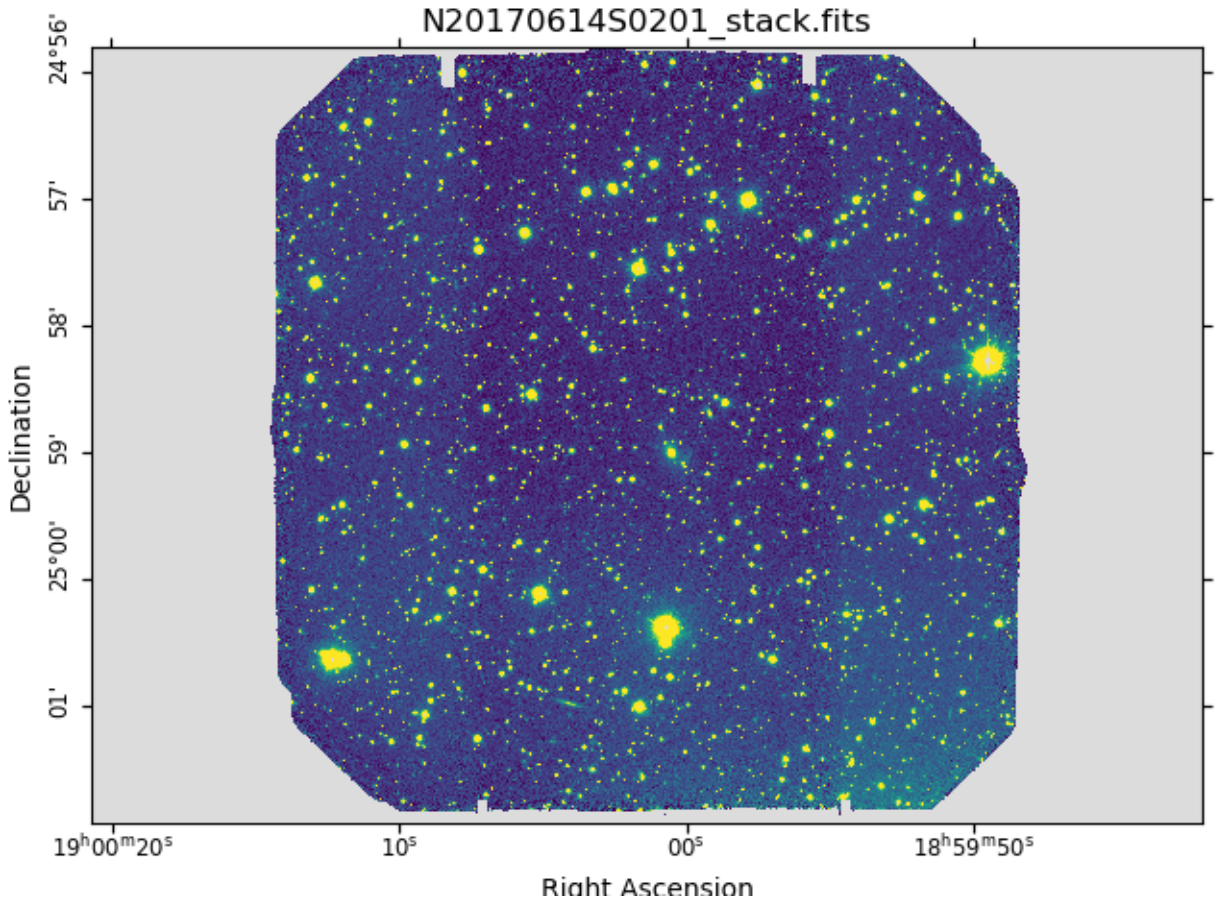

Fig. 2: Final stacked image. The light-gray area represents the masked pixels.

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	N20170614S0201-205	10 s, i-band
Bias	N20170613S0180-184 N20170615S0534-538	
Twilight Flats	N20170702S0178-182	40 to 16 s, i-band

3.2 Setting Up

3.2.1 Importing Libraries

We first import the necessary modules and classes:

```
import glob

from gempy.adlibrary import dataselect
from recipe_system import cal_service
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*.)

```
from gempy.utils import logutils
logutils.config(file_name='gmos_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/gmosimg_tutorial_api/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 Using the caldb API in the Recipe System User's Manual .

3.3 Create list of files

The 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/`.

```
15 all_files = glob.glob('../playdata/*.fits')
16 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 List of Biases

Let us, now, select the files that will be used to create a master bias:

```
17 list_of_biases = dataselect.select_data(
18     all_files,
19     ['BIAS'],
20     []
21 )
```

Note the empty list `[]` in line 20. This positional argument receives a list of tags that will be used to exclude any files with the matching tag from our selection (i.e., equivalent to the `--xtags` option).

3.3.2 List of Flats

Next we create a list of twilight flats 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.

```
22 list_of_flats = dataselect.select_data(
23     all_files,
24     ['FLAT'],
25     [],
26     dataselect.expr_parser('filter_name=="i"')
27 )
```

3.3.3 List of Science Data

Finally, the science data can be selected using:

```
27 list_of_science = dataselect.select_data(
28     all_files,
29     [],
30     ['CAL'],
31     dataselect.expr_parser('(observation_class=="science" and filter_name=="i")')
32 )
```

Here we left the `tags` argument as an empty list and passed the tag `'CAL'` as an exclusion tag through the `xtags` argument.

We also added a fourth argument which is not necessary for our current dataset but that can be useful for others. It contains an expression that has to be parsed by `expr_parser()`, and which ensures that we are getting *science* frames obtained with the *i-band* filter.

3.4 Make Master Bias

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

```
33 reduce_bias = Reduce()
34 reduce_bias.files.extend(list_of_biases)
35 reduce_bias.runr()
36
37 caldb.add_cal(reduce_bias.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.

Once `runr()` is finished, we add the master bias to the calibration manager (line 37).

3.5 Make Master Flat

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

```
38 reduce_flats = Reduce()
39 reduce_flats.files.extend(list_of_flats)
40 reduce_flats.runr()
41
42 caldb.add_cal(reduce_flats.output_filenames[0])
```

3.6 Make Master Fringe Frame

Warning: The dataset used in this tutorial does not require fringe correction so we skip this step. To find out how to produce a master fringe frame, see *Create Master Fringe Frame* in the *Tips and Tricks* chapter.

3.7 Reduce Science Images

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

```
43 reduce_science = Reduce()
44 reduce_science.files.extend(list_of_science)
45 reduce_science.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 Create Master Fringe Frame

The reduction of some datasets requires a master fringe frame. The filters that need a fringe frame are shown in the appendix *Fringe Correction Tables*.

To create the master fringe frame from the dithered science observations and add it to the calibration database:

```
$ reduce @list_of_science.txt -r makeProcessedFringe
$ caldb add N20170614S0201_fringe.fits
```

This command line will produce an image with the `_fringe` suffix in the current working directory.

Again, note that this step is only needed for images obtained with some detector and filter combinations. Make sure you checked the *Fringe Correction Tables*.

The above can be done with the API as follows:

```
reduce_fringe = Reduce()
reduce_fringe.files.extend(list_of_science)
reduce_fringe.recipename = 'makeProcessedFringe'
reduce_fringe.runr()

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

4.2 Bypass 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_bias:S20001231S0001_bias.fits␣
↪processed_flat:S20001231S0002_flat.fits
```

The list of recognized processed calibration is:

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

4.3 Browse Recipes and Primitives

"", either the command line or the API class, is the tool that selects and run a "recipe". A recipe is a sequence of operations called "primitives". Each primitives has a defined set of input parameters with default values that can be overriden by the user.

The "" command line is used to show the default recipe for a file, a specific recipe for that file, or all the recipes associated with the file.

Once you know the recipe and primitives it is calling, you can explore the primitives' parameters using the "" command line.

The tools are fully documented in the .

4.4 Customizing input parameters

From the command line, setting the value of a primitive input parameter is done as follow:

```
$ reduce @sci.lis -p stackFrames:scale=True
```

The `-p` flag indicates that the following items are parameter changes. The syntax is `<primitive_name>:<parameter_name>=<value>`

From the API, the `uparms` attribute to the `Reduce` instance is used.

```
reduce_science.uparms.append(("stackFrames:scale", True))
```

4.5 Setting the output suffix

When troubleshooting an issue or trying various settings to optimize a reduction, it might be useful to name the final recipe output differently for each attempt.

Only the **suffix** of the final output file can be changed, not its full name.

From the command line:

```
$ reduce @sci.lis --suffix='newsuffix'
```

From the API:

```
reduce_science.suffix = "newsuffix"
reduce_science.runr()
```

CHAPTER 5

Issues and Limitations

5.1 Memory Usage

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 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='gmos_data_reduction.log')
```

5.3 Astropy warnings

You might see some warning messages from AstroPy that are related to the header of the images. It is safe to ignore them.

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 a Science Verification program done during the commissioning and characterizing phase of the GMOS-N Hamamamatsu CCDs. The program ID is GN-2017A-SV-151.

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

Access the Gemini Observatory Archive (GOA) and fill the search form as follow:

- Program ID: GN-2017A-SV-151-382
- Instrument: GMOS-N
- Filter: i'

Press the `Search` button in the middle of the page.

The table will show you 10 files. Mark the checkbox for the first 5 files in the list. Normally, you would use all 10 files, but for the purpose of the tutorial, 5 files will do and will run faster.

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

```
https://archive.gemini.edu/searchform/GN-2017A-SV-151-382/cols=CTOWEQ/filter=i/
↪notengineering/GMOS-N/imaging/science/NotFail
```

6.1.2 Calibrations

The calibration files could be obtained by simply clicking on the **Load Associated Calibrations** tab. You will see that the Gemini Archive will load much more files than we need (129 files, totalling 0.53 Gb). Obviously we don't need all that.

For this data, we need a few biases and a few twilight flats, all taken around the time of the observations. How many to download depends on your personal philosophy to some extend. For the biases, using 10 to 20 raw biases works well. For the twilight flats, make sure that they are set to "Pass", do not use the "Usable" if you can avoid it. In this case, because it was commissioning data, the quality status was not set and all calibrations are set to "Undefined". It will be fine for our purpose.

For this tutorial, we will pick the 10 biases taken on the day previous to our observations since none were taken on the day. The twilight flats from 2017 July 2, GN-CAL20170702-3, are the closest in time to our observations, we will use those.

For the biases, let's pick the first ten (10) on the list, skipping the very top one which comes from an engineering program (the GN-ENG- in the program ID gives it up). The selected biases are from observation ID GN-CAL20170613-3 and GN-CAL20170615-14. Select the checkboxes on the left.

For the twilight flats, scroll down the table until you see them, about half way down. Be mindful of the last column, we normally must select the flats with a "Pass" status. Here all the flats are set to "Undefined" because this was commissioning data so we will have to make due with them. Let's pick the flats from the night of 2017 July 2 with observation ID GN-CAL20170702-3. Let's pick the first 5 flats. Select them checkboxes on the left.

Now scroll all the way down and press the "Download Marked Files" button.

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 for running the tutorial called `playground`. The tutorial makes assumption as to where everything is located.

CHAPTER 7

Fringe Correction Tables

Here you will find what are the detector-filter combinations that requires a Processed Fringe Frame for the data reduction. Below are one table for GMOS-N and one table for GMOS-S. Each row of these tables corresponds to one of the detectors used in the instrument during its life-time. The five columns in the right contains the broadband filters used in imaging mode. The intersection of the detector rows with the filter columns contains cell with the following information:

- *Yes*: Requires a Processed Fringe Frame for data reduction;
- *No*: Does not require a Processed Fringe Frame for data reduction;
- —: Does not have data with these filters.

Table 1: GMOS-N Table for Detector/Filter Configurations that require Processed Fringe Frame for data reduction. Filters bluer than i' do not require fringe correction.

GMOS-N		i'	CaT	Z	z'	Y
EEV CCDs	Aug 2001 - Nov 2011	Yes	Yes	—	Yes	—
E2V DD CCDs	Nov 2011 - Feb 2017	Yes	Yes	Yes	Yes	Yes
Hamamatsu CCDs	February 2017 - Present	No	No	No	Yes	Yes

More: GMOS-N Fringe Information

Table 2: GMOS-S Table for Detector/Filter Configurations that require Processed Fringe Frame for data reduction. Filters bluer than i' do not require fringe correction.

GMOS-S		i'	CaT	Z	z'	Y
EEV CCDs	Commissioning - June 2014	Yes	Yes	—	Yes	—
Hamamatsu CCDs	June 2014 - Present	No	No	No	Yes	Yes

More: GMOS-S Fringe Information

CHAPTER 8

Indices and tables

- genindex
- modindex
- search