# Collection of DRAGONS Workshops

***Release 1.1***

**Kathleen Labrie**

**April 2023**

# CONTENTS:


**1 Basic DRAGONS Workshop** **1**
- 1.1 Learning Objectives . . . 1
- 1.2 What is DRAGONS . . . 2
- 1.3 What are Recipes and Recipe Libraries . . . 4
- 1.4 What are primitives . . . 6
- 1.5 Demo imaging . . . 8
- 1.6 Explore data . . . 11
- 1.7 Local calibration manager . . . 13
- 1.8 Using `reduce` . . . 15
- 1.9 Customize Recipes . . . 19
- 1.10 Tools . . . 21
- 1.11 Resources . . . 25
- 1.12 Solutions to exercises . . . 25

**2 Basic DRAGONS Workshop - Short Version** **35**
- 2.1 Learning Objectives . . . 35
- 2.2 What is DRAGONS . . . 36
- 2.3 What are Recipes, Recipe Libraries, and Primitives . . . 38
- 2.4 Demo imaging . . . 44
- 2.5 Explore data . . . 47
- 2.6 Local calibration manager . . . 47
- 2.7 Using `reduce` . . . 50
- 2.8 Customize Recipes . . . 52
- 2.9 Tools . . . 54
- 2.10 Resources . . . 58
- 2.11 Solutions to exercises . . . 58

**3 Indices and tables** **65**

CHAPTER

ONE

# BASIC DRAGONS WORKSHOP

## 1.1 Learning Objectives

This material is built for a 2-hr workshop, planning on 90 minutes for the workshop and 30 minutes for questions and helping participants.

In this workshop, the participants will gain a basic understanding about the following topic:

- What is DRAGONS, what it isn't.
- What is a typical DRAGONS workflow.
- What are the principal components of DRAGONS.
- What is a recipe, a recipe library, and a primitive.
- How to do coarse inspection of the data.
- How to use the calibration manager.
- How to customize input parameters and recipes.
- How to control `reduce`.
- What utility tools are available and how to use them.

This workshop tries to cover a lot in a short period of time. The focus will be on some key, often needed elements. The participants should refer to the DRAGONS documentation for more in-depth information.

This workshop uses the command-line interface to DRAGONS. An API is available but will not be covered in this workshop.

### 1.1.1 Setting up

The participants who wish to following along and run the examples and exercises are expected to have already installed DRAGONS on their computer. See the **installation instructions**, .

Follow the Python 3 and DRAGONS instructions, IRAF is not needed. Make sure that you do the steps in the "Configure DRAGONS" section.

---

**Important: This workshop is written for DRAGONS v3.1**

---

The participants will need to download the data package to run the examples and the exercises:

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

Download it and unpack it somewhere convenient.

```
cd <somewhere convenient>
tar xvf niriim_tutorial_datapkg-extended-v1.tar
bunzip2 niriimg_tutorial/playdata/example1/*.bz2
```

The datasets are found in the subdirectory `niriimg_tutorial/playdata/example1/`, and we will work in the subdirectory named `niriimg_tutorial/playground`.

---

**Note:** If you are attending a live workshop, it is advisable to also download the preprocessed data package in case you encounter technical difficulties that prevents you from running the full demo yourself. The content of that package will allow you to continue with most of the exercises using those preprocessed data. Download the "just-in-case" package at:

https://drive.google.com/file/d/1sJd8bHUFZ3-dNVT3_VllcqplnoSZr4LJ/view?usp=sharing

You do not need to do anything with it now other then keeping it somewhere safe. Hopefully, everything will run smoothly and you will not need it.

---

## 1.2 What is DRAGONS

### 1.2.1 What it is

DRAGONS is a platform for the reduction and processing of astronomical data. It is a Python meta-package that includes infrastructure for automation and algorithms for processing astronomical data. Currently, it focuses on the reduction of Gemini data, but hooks are available for expansion to data from other observatories.

Now really, what is DRAGONS?

With a little bit of guidance from you, DRAGONS will reduce your Gemini data in a standard way, rapidly, with little fuss. With a bit more guidance from you, DRAGONS will let you optimize the reduction for your specific sources and science objectives.

DRAGONS is operated through the `reduce` command from the shell, or the `Reduce` class from a Python script. You choose. For clarity, we will refer to `reduce` only from now on, but it's the same thing. A call to *reduce* activates the "Recipe System" which is what automates everything.

When `reduce` is called, the first FITS file is opened and identified, then the libraries of algorithms collected as "primitives" and "recipes" will be searched and the best matches will be selected and run on the data.

The FITS files are opened with the `astrodata` facility. When a FITS file is opened with `astrodata`, it "knows about itself". For example, it knows that it is a NIRI file and an IMAGING FLAT. This is how the Recipe System will know which recipes and which primitives to load and run.

The other important component of the DRAGONS automation infrastructure is the calibration manager. It is operated with the `caldb` command. (Again, there's an API too.) This accesses a lightweight local `sqlite` database that will store information about your locally-processed calibrations. When a primitive is run and needs a calibration, the system will automatically identify the best matched processed calibration and use it, you do not have to specify it on the command line. (Though, you can if you really want to.)

Then there's a series of utilities for sorting through the data and learning about the primitives and the recipes.

Here's what DRAGONS can look like from a user's point view:

```
dataselect *.fits --tags=FLAT -o flat.lis
dataselect *.fits --expr="object='mycooltarget'" -o sci.lis
```

(continues on next page)

(continued from previous page)

```
caldb init
reduce @flat.lis
caldb add *_flat.fits
reduce @sci.lis
```

The steps are:

1. Create your lists of input data.
2. Initialize your calibration database.
3. Reduce your calibrations and upload the info to the database (the upload can be automated).
4. Reduce your science, it will pick up the calibrations by itself.

As straightforward as a reduction can be, it can be customized to match your needs. The options to the primitives can be adjusted, the recipes themselves can be adjusted.

In this basic introduction to DRAGONS, we will explore all this and learn how to have DRAGONS do our bidding.

### 1.2.2 What it is not

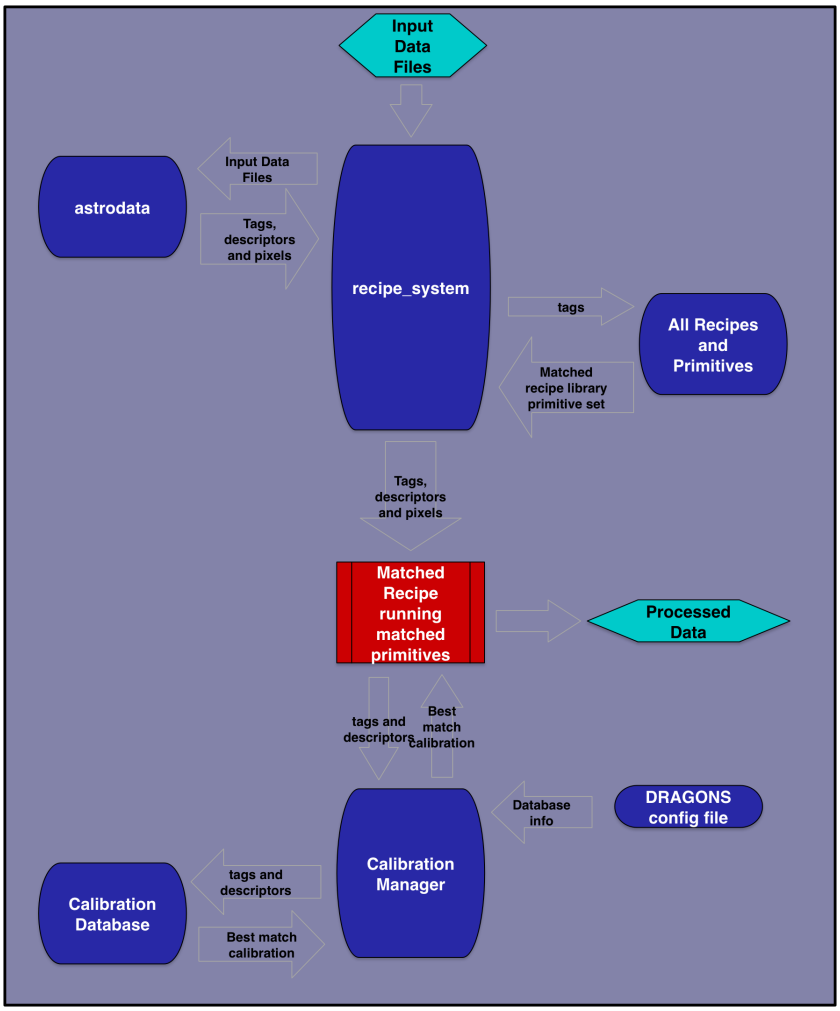
DRAGONS is **not** a data analysis package. DRAGONS prepares the data for analysis but does not offer tools to do the analysis. Also, DRAGONS is also not a replacement for IRAF.

To inspect your data, to do any sort of analysis like photometry, measuring redshift, calculating equivalent width, etc, you will have to find another tool. DRAGONS does provide a primitive to display an image to ds9, but it will just display it. Similarly, there is a tool to plot a 1D spectrum, and again, it will just plot it, it does not offer any measurement capabilities.

Any analysis-type tools are beyond the scope of DRAGONS.

### 1.2.3 Components in action

The diagram below illustrates how the components communicate with each other and in particular how the `astrodata` tags and descriptors are passed around to make decisions about the best matched recipe, primitive sets, and calibrations.




## 1.3 What are Recipes and Recipe Libraries

A DRAGONS **recipe** is a set of instructions, called **primitives**, that processes data in a certain way. Depending on the recipe, the input data can be in any state of processing. Most often though, people will want to process Gemini raw data into a master calibration or a processed calibrated output.

A **recipe library** is a collection of recipes. The recipes in a library have one thing in common: *they all apply to the same type of data*. When DRAGONS search for a matching recipe for some input data, it searches for a recipe library. In each library, one recipe is set as the default recipe. To use the others, the user needs to specify the name of the non-default recipes.

Let's look at examples and tools.

### 1.3.1 Recipes

This is what a recipe looks like:

```
def reduce(p):
   p.prepare()
   p.addDQ()
   p.removeFirstFrame()
   p.ADUToElectrons()
   p.addVAR(read_noise=True, poisson_noise=True)
   p.nonlinearityCorrect()
   p.darkCorrect()
   p.flatCorrect()
   p.separateSky()
   p.associateSky(stream='sky')
   p.skyCorrect(instream='sky', mask_objects=False, outstream='skysub')
   p.detectSources(stream='skysub')
   p.transferAttribute(stream='sky', source='skysub', attribute='OBJMASK')
   p.clearStream(stream='skysub')
   p.associateSky()
   p.skyCorrect(mask_objects=True)
   p.detectSources()
   p.adjustWCSToReference()
   p.resampleToCommonFrame()
   p.stackFrames()
   p.writeOutputs()
```

A recipe is a Python function that calls "primitives" sequentially. The user does not need to know Python to understand more or less what is being done to the data when the recipe is run. From the recipe above, we can read that the data will be corrected for dark current, for flat field effects, the sky background will be subtracted and the frames will be stacked. The *p* argument is the **primitive set** that matches the input. We will talk about that in the next chapter.

#### Exploring recipes

can be used to see which recipe DRAGONS with run by default given the input data. The tool's basic usage signature is:

```
showrecipes fitsfile_name.fits
```

To see more options, try `showrecipes -h`.

---

**Exercise - Recipes 1**

The file `N20160102S0373.fits`, in the `playdata/example1` directory, is a raw "lamp-on" flat.

- Get the default recipe for a NIRI flat displayed on the terminal
- Find the name of the default recipe.

[*Solution*]

---

### 1.3.2 Recipe libraries

Recipes are stored in recipe libraries, or in Python language, a module. A recipe library can have one or more recipes in them. One recipe is identified as the default recipe.

When DRAGONS searches for a recipe, it is actually searching for a matching recipe library. Once found, it will run the default recipe, unless instructed otherwise by the user.

The recipe libraries are associated with the input files by matching `astrodata` tags (). The tags are qualifiers like "NIRI", "IMAGE", "FLAT". The tags of the first file in the list of inputs are used by the matching algorithm. Each recipe library is assigned a set of tags that defines which type of data this library is for.

---

**Exercise - Recipes 2**

The file `N20160102S0423.fits` in the `playdata/example1` directory is a raw dark.

- Find the location of the recipe library on your disk.
- What are the tags assigned to the input file?
- What are the tags assigned to that recipe library?

[*Solution*]

---

As mentioned above, a recipe library can have more than one recipe. To see the list of recipes, has the option `--all`. Then, to see the sequence of primitives for one of those recipes, there's the option `-r`.

DRAGONS has the concept of "reduction mode". There are three modes: the science quality mode, "sq", the quicklook mode, "ql", and the quality assessment mode, "qa". You can safely ignore the "qa" mode, it is used exclusively at the observatory, at night, to help with the assessment of the sky conditions and the resulting quality of the data. The mode can be specified in with the `-m` flag. The default is "sq".

To see all the flags and option, `showrecipes -h`.

---

**Exercise - Recipes 3**

For the file `N20160102S0270.fits` in the `playdata/example1` directory:

- List all matching recipes. Note the "sq", and the "qa" recipes.
- Show the `makeSkyFlat` recipe. ("sq" is the default.)
- Show the `reduce` recipe for "qa" mode.

[*Solution*]

---

## 1.4 What are primitives

### 1.4.1 Primitives and primitive sets

A primitive is a data reduction step involving a transformation of the data or providing a service. By convention, the primitives are named to convey the scientific meaning of the transformation. For example `biasCorrect` will remove the bias signal from the input data.

A primitive is always a member of a primitive set. It is the primitive set that gets matched to the data by the Recipe System, not the individual primitives.

Technically, a primitive is a method of a primitive class. A primitive class gets associated with the input dataset by matching the `astrodata` tags. Once associated, all the primitives in that class, locally defined or inherited, are available to reduce that dataset. We refer to that collection of primitives as a "primitive set".

All the primitives are currently found in the Python package `geminidr`. In there, there is a large inheritance tree of primitive classes from very generic to very specific.

For example, NIRI images will be associated with the `NIRIImage` primitive set. This set contains all the primitives it needs through primitives defined in `NIRIImage` itself and through its inheritance tree. We illustrate that tree in Figure 1.

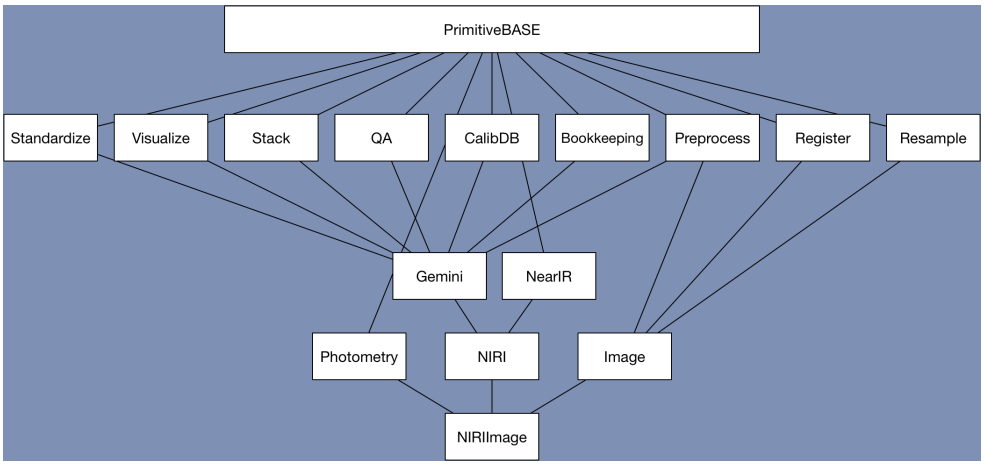



Fig. 1: Figure 1: NIRIImage inheritance tree.

Once the primitive set is identified, DRAGONS will send that to the selected recipe as that `p` argument we saw in the previous chapter.

### 1.4.2 Primitive input parameters

Also attached to a primitive set are the input parameters for each primitives and the defaults appropriate for that primitive set, that is for that type of data.

For the same generic primitive, the default values for input parameters for NIRI data can be different from the defaults applicable to GMOS data. Even the set of available input parameters can be different, though we try to keep things as uniform as possible.

We will learn later how to customize the values of input parameters. The first step however is to see what the input parameters are and what their default values are. This is done with the command.

Because the input parameters can be different for different type of data, the command requires the name of file, and then of course the name of the primitive you want to learn about.

The command signature is as follow:

```
showpars filename primitive_name
```

---

**Exercise - Primitives 1**

Using the science frame `N20160102S0270.fits` located in directory `playdata/example1`, find the answer to the questions below. First, show the recipe for that file to see the list of primitives. (Hint: `showrecipes`)

1. What is the default operation used to combine the sky frames during sky correction?

2. Which parameter from which primitive do I need to modify to turn off the dark correction?
3. Which parameter from which primitive do I need to modify to change the rejection method during the final stacking of the frames? What are the available options?

[*Solution*]

## 1.5 Demo imaging

To get ourselves oriented and get a feel for how data is processed with DRAGONS, we will run a full reduction on our sample data. The full tutorial for reducing NIRI data can be found at .

Here, we go through the steps with a bit less discussion than in the full tutorial as the purpose is simply to provide some context and an idea of a typical data reduction flow to those who have not experience it before.

The observations are of an extended source, a nearby galaxy. The instrument used is NIRI. The science sequence is a series of dithers on target with full offset to sky. We will create a master dark, a master flat, a reduced flux standard, and finally create a calibrated, sky-subtracted stack of the science observations.

Let's run all the steps.

1. Set up the local calibration manager
2. Create the file lists
3. Create the calibration files (dark, flat)
4. Reduce the flux standard
5. Reduce the science observations

```
cd <where_the_data_package_is>/niriimg_tutorial/playground
```

### 1.5.1 Set up the local calibration manager

We will discuss the local calibration manager in a later chapter.

The calibration manager is the first thing to set up when starting on a data reduction project. It provides the automated calibration association.

The file to pay attention to is: `~/.dragons/dragonsrc` (DRAGONS v3.1+)

Edit that file to contain the following:

```
[calibs]
databases = <where_the_data_package_is>/niriimg_tutorial/playground/cal_manager.db get
```

In this configuration, with the "get" flag, DRAGONS will retrieve the processed calibration automatically from that database. You could add "store" after the "get" to have DRAGONS add the processed calibrations to the database automatically (effectively running `caldb add` for you). Here we will add the calibrations it manually.

About the path for `databases`, we recommend the directory you are using to work on the data. But it can be anywhere.

Once the configuration file is set up, initialize the database. This needs to be done only once per database.

```
caldb init
```

This creates a new file, `cal_manager.db`, in the directory specified by `databases`.

### 1.5.2 Create the file lists

The user has to sort the files. DRAGONS is a "User-in-the-Loop" pipeline. We have some tools to help. We will have a closer look at them later.

We have long darks that match the science, a series of lamp-on and lamp-off flats, a set of on-target dithered observations for a flux standard, and a set of on-target dithered and off-target dithered observations for the science target.

```
dataselect ../playdata/example1/*.fits --tags DARK --expr='exposure_time==20' -o␣
↪darks20s.lis

dataselect ../playdata/example1/*.fits --tags FLAT -o flats.lis

dataselect ../playdata/example1/*.fits --expr='object=="FS 17"' -o stdstar.lis

dataselect ../playdata/example1/*.fits --expr='object=="SN2014J"' -o target.lis
```

### 1.5.3 Add static BPM to calibration database

The static BPMs are downloaded from the archive (but included here in our data package).

```
caldb add ../playdata/example1/bpm_20010317_niri_niri_11_full_1amp.fits
```

### 1.5.4 Create the master dark and the master flat

**Dark**

Create the master dark and add it to the calibration manager.

```
reduce @darks20s.lis
caldb add N20160102S0423_dark.fits
```

The @ character before the name of the input file is the "at-file" syntax. We will look into this later.

**Flat**

Create the master flat from the lamp-on and lamp-off flats and add it to the calibration database.

```
reduce @flats.lis
caldb add N20160102S0373_flat.fits
```

### 1.5.5 Reduce the flux standard

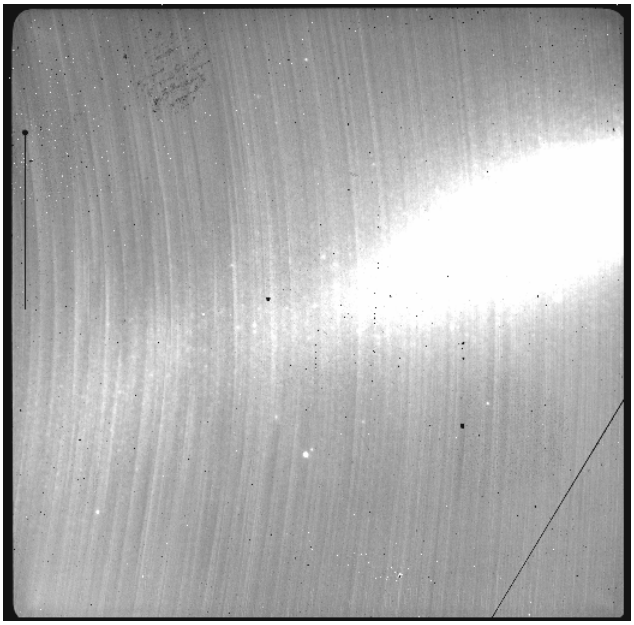
This is short series of on-target dither observations. Darks are not obtained for NIRI flux standard so we turn that step off. The flat will be picked up from the calibration database.

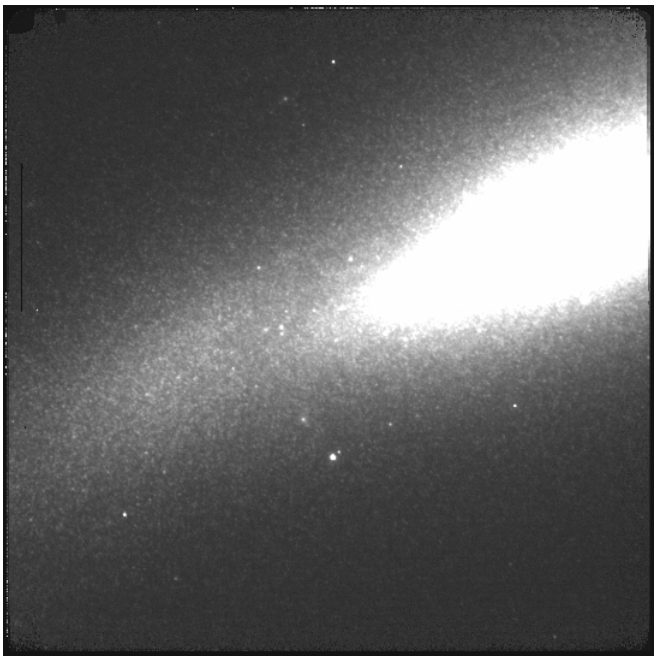
```
reduce @stdstar.lis -p darkCorrect:do_cal=skip
```

### 1.5.6 Reduce the science observations

The sequence is a set of dither on target with full offsets to sky. DRAGONS will sort them during the sky correction. All calibrations are picked up automatically from the calibration database. Because the target fills the field of view, we need to turn off the scaling of the sky frames.

```
reduce @target.lis -p skyCorrect:scale_sky=False
```





## 1.6 Explore data

Once we have reduced our data, we probably want to look at. In fact, even before we start the reduction, we should look at the raw data in case of issues.

DRAGONS is **not** an analysis package. But it does offer a basic way to display images.

The `display` and `inspect` primitives can be used to visually inspect the data with `ds9`. Those are primitives not recipes, yet `reduce` can run individual primitives.

---

**Note:** `ds9` must be launched by the user ahead of running the primitive.

---

### 1.6.1 Display

To display an image to `ds9`:

```
reduce -r display N20160102S0271_stack.fits
```

By default, the saturated pixels are marked. This can be turned off by setting the `threshold` parameter to "None".

### 1.6.2 Inspect

It is recommended to inspect the raw data prior to reducing the data. A quick visual inspection can catch obviously bad data, or flag some frames for more detailed inspection. Having to run display on each frame individually would be painful. The primitive `inspect` can be used here. It accepts a list of file and will display each frame in sequence, adding a short pause between the frames. The length of the pause can be controlled with the `pause` parameter.

```
reduce -r inspect @stdstar.lis
```

---

**Exercise - Explore 1**

1. Display `N20160102S0271_stack.fits` to buffer 2 and turn off the saturation marking.
2. Reduce the time between frames to 1 second in `inspect`.

Hint: `reduce file.fits -r primitive -p primitive:parameter=value`

[*Solution*]

---

### 1.6.3 Accessing table content

Some output files will have table information. Once written to disk as a FITS file, the tables are accessible as any FITS files. A common table is a list of sources detected by the primitive `detectSources`.

Let say that we want to get position of the sources in the flux standard stack we produced in the previous chapter. First, we run `detectSources` since it was not run after the stack was created.

```
reduce -r detectSources N20160102S0296_stack.fits
```

This will add a table with object coordinates, magnitudes, etc. The output file is `N20160102S0296_sourcesDetected.fits`. Use your preferred tool to access the FITS table named `OBJCAT` from that output. Or you can use DRAGONS' `astrodata` to access the table as an Astropy `Table`.

```
import astrodata
import gemini_instruments

ad = astrodata.open('N20160102S0296_sourcesDetected.fits')
ad.info()
```

```
Filename: N20160102S0296_sourcesDetected.fits
Tags: GEMINI IMAGE NIRI NORTH PREPARED SIDEREAL

Pixels Extensions
Index  Content                Type              Dimensions     Format
[ 0]   science                NDAstroData       (1195, 1195)   float32
          .variance           ndarray           (1195, 1195)   float32
          .mask               ndarray           (1195, 1195)   uint16
          .OBJCAT             Table             (29, 43)       n/a
          .OBJMASK            ndarray           (1195, 1195)   uint8
```

```
ad[0].OBJCAT
```

```
<Table length=29>
NUMBER  X_IMAGE   Y_IMAGE        ERRX2_IMAGE       ... REF_MAG REF_MAG_ERR PROFILE_FWHM␣
↪PROFILE_EE50
int32   float32   float32          float64         ... float32   float32     float32    ␣
↪ float32
------ --------- ---------- ---------------------- ... ------- ----------- ------------ -
↪-----------
     1 638.74554   7.358276  9.678123990684229e-06 ...  -999.0      -999.0       -999.0 ␣
↪     -999.0
     2   52.5378  15.138286  0.0007821264390706773 ...  -999.0      -999.0          0.0 ␣
↪  3.5255525
     3 500.59454  1146.9167   0.012262252608200411 ...  -999.0      -999.0     4.513517 ␣
↪  6.1514697
     4 879.47473  1098.4614  2.762548642098426e-05 ...  -999.0      -999.0     4.222008 ␣
↪  5.0135765
     5  99.23588  1012.4829  7.044837003541005e-06 ...  -999.0      -999.0    3.5682483 ␣
↪  4.5748672
     6   67.8556 1023.36334  0.0036609541234138973 ...  -999.0      -999.0    3.7424104 ␣
↪  4.9484925
     7   654.096  918.19916 2.5297123639012174e-05 ...  -999.0      -999.0    4.6524267 ␣
↪   4.829588
     8 855.02795   830.9594  6.732778544641382e-06 ...  -999.0      -999.0     6.675581 ␣
↪  6.2657185
     9 1026.4033    835.157     0.0433908185530153 ...  -999.0      -999.0     4.068429 ␣
↪  12.851359
    10  380.1143  726.31464 0.0004170915641706719 5 ...  -999.0      -999.0     4.370194 ␣
↪  4.4451947
    11 994.73755    721.932   0.012686314238205688 ...  -999.0      -999.0    4.6524267 ␣
↪    9.30962
   ...       ...        ...                    ... ...     ...         ...          ... ␣
↪        ...
    18  854.6116  352.42896 1.5660364975379077e-05 ...  -999.0      -999.0     4.222008 ␣
↪   4.193287
```

(continues on next page)

(continued from previous page)

```
    19 522.56934  317.01556  0.0012430236331253412 ...  -999.0      -999.0    3.5682483 ␣
↪   3.738548
    20 936.59576  242.70079  7.264929326425364e-06 ...  -999.0      -999.0     4.068429 ␣
↪  4.1155357
    21  811.8014  225.92892   8.23835737153326e-05 ...  -999.0      -999.0    3.9088202 ␣
↪  4.1611533
    22 478.92426  217.57622    0.02797088088581828 ...  -999.0      -999.0     8.812923 ␣
↪     -999.0
    23  479.3753   206.3997   0.022581652547939684 ...  -999.0      -999.0    12.257336 ␣
↪  12.230416
    24 478.64758  194.86026   0.009814622938623417 ...  -999.0      -999.0    2.7639532 ␣
↪  15.229653
    25 521.57794  209.82019   0.029336651973282495 ...  -999.0      -999.0    3.1915383 ␣
↪  16.864668
    26  479.6435  152.71979 3.3380840555892644e-06 ...  -999.0      -999.0    3.9088202 ␣
↪  3.7304442
    27 955.26416   150.0823  7.081766055952015e-05 ...  -999.0      -999.0    3.7424104 ␣
↪   4.108046
    28 561.57697  151.73418   0.021476893612064937 ...  -999.0      -999.0     6.675581 ␣
↪   9.169356
    29 129.58281  107.06496  0.0009807660866281128 ...  -999.0      -999.0    3.3851376 ␣
↪  3.6767333
```

See the `astropy` documentation for information about how to operate on `Table` objects: https://docs.astropy.org/en/stable/table/.

For more on `astrodata`, see the .

## 1.7 Local calibration manager

The local calibration manager is the tool that will allow DRAGONS to find the most appropriate processed calibration for the data being reduced.

The local calibration manager contains all the same rules as the Gemini Observatory Archive, but works locally, with your files, without internet, and alongside DRAGONS to make the calibration association as seamless as possible. As we have seen in the demo earlier, we create the master calibrations, we add them to the database, and when DRAGONS (`reduce`) needs one, it gets matched and retrieved automatically.

The system uses a lightweight `sqlite` database that will **not** store the data file, it will store only information **about** the file, like its location on disk, and key astrodata tags and descriptors.

> **Warning:** If you move or delete a processed calibration on disk after having entered it in the database, the calibration manager will no longer be able to find it. The database does not contain the file, just information about the file.

See the documentation for more information, including information about the programming interface.

### 1.7.1 Configuration and initialization

**Warning:** The information provided here applies to DRAGONS version 3.1 (and newer). The configuration and features of the calibration manager in the previous versions, 3.0 and older, are substantially different. Please refer to a previous version of this workhop if you are using DRAGONS 3.0 or older,

https://collection-of-dragons-workshops.readthedocs.io/en/v1.0.0/BasicDRAGONS-2hr/sphinx/caldb.html

#### Configuration

The behavior and configuration of the local calibration manager are controlled in the file `~/.dragons/dragonsrc`. When you first install DRAGONS, you will have to create that directory and that file.

If you ran the demo earlier, the `dragonsrc` file should look like this:

```
[calibs]
databases = <where_the_data_package_is>/niriimg_tutorial/playground/cal_manager.db get
```

The value of `databases` can be any **existing** path on your machine. The calibration database will be created, and then expected to be in that directory. The database name can be anything, here we name it `cal_manager.db` for continuity with pre-v3.1 DRAGONS.

We recommend setting up a new database for each observing project. It helps keep things clean and avoids calibrations from other programs being unexpectedly picked up. (If they were picked up, they would be a match, but it might get confusing if you are expecting another calibration to be picked up.)

Here is an example of a `dragonsrc` with several `databases` entries. Note that at all times, **only one** `databases` directive is active, the others are commented out.

```
[calibs]
databases = /Users/klabrie/data/tutorials/niriimg_tutorial/playground/cal_manager.db get
#databases = /Users/klabrie/data/tutorials/gmosimg_tutorial/playground/cal_manager.db get

#databases = /Users/klabrie/data/workspace/GMOS540feature/cal_manager.db get
```

#### Initialization

Once your `dragonsrc` is set up, you can create a **new** database with

```
caldb init
```

If the database already exists, that command will refuse to work. This ensures that you will not wipe the database accidentally. **The `init` is required only once.** If you already have the database created, maybe even populated, you can switch back to it simply by ensuring `databases` points to it.

---

**Exercise - Caldb 1**

Create a new database in the `niri_tutorial` directory. Keep the `playground` entry that should already be there, but deactivate it.

Confirm with `ls` that a new `cal_manager.db` has been created. You can also use `caldb config` to confirm that the new database is active.

[*Solution*]

### 1.7.2 Usage

The usage is fairly basic. One can `list` the content, `add` content, and `remove` content. DRAGONS will retrieve content from the database. If the "store" flag is set, DRAGONS will also automatically add content to the database.

> **Warning:** In DRAGONS version 3.0 and older, DRAGONS cannot add content automatically, the user **must** add the processed calibrations to the database with an explicit `caldb add`.

To verify which database is currently active, use the `config` option:

```
caldb config
```

To see the content of the database:

```
caldb list
```

To add information about a processed calibration to the database:

```
caldb add name_of_file.fits
```

And finally, to delete information about one file from the database:

```
caldb remove name_of_file.fits
```

`remove` will only remove the entry in the database, it will not remove the file on disk.

**Exercise - Caldb 2**

Do *Exercise - Caldb 1* first.

1. Add the flat field from the demo to the new calibration manager created in the first exercise.
2. Show the content of that database is indeed just that file.
3. Reactivate the original database (the one in `playground` that we used for the demo) and list its content. Both the dark and the flat should now be listed.

[*Solution*]

## 1.8 Using `reduce`

The `reduce` command (and the `Reduce` class if you are using the programming interface) is what drives DRAGONS. The command has a lot of options. We have used a few already. Let's review those and show a few more.

First, to see the list of options available, use the `-h` flag:

```
reduce -h
```

### 1.8.1 Setting the suffix of the final outputs

It is not possible to set the name of the final output. This seems odd, but here is why. There might be more than one output, there can be dozens, it depends on the recipe.

We are still exploring solutions to this problem.

In the meantime, the user can set the **suffix** of the outputs. Each primitive has a meaningful suffix attached to it, for example `_sourcesDetected` for the primitive `detectSources`. The output of a `reduce` call will get the suffix assigned to the last primitive of the recipe.

If you wish to override that, you can set the `--suffix` option.

```
reduce @targets --suffix=_mytest
```

---

**Exercise - "reduce" 1**

Run the primitive `detectSources` on the flux standard stack (`N20160102S0296_stack.fits`) and set the output suffix to `_ILoveDRAGONS`.

Hint: To specify a recipe or a primitive to run, remember the *-r* flag.

[*Solution*]

---

### 1.8.2 Customizing parameters

We have seen earlier how to check the available input parameters and their default settings for a given frame and primitive. The command does that. For example:

```
showpars ../playdata/example1/N20160102S0363.fits normalizeFlat
```

The syntax to change the value of an input parameter is as follow:

```
reduce file.fits -p primitive:parameter=value
```

---

**Exercise - "reduce" 2**

Reduce the flats from the demo again but this time set the scaling for `normalizeFlat` flat to `mean`. To avoid overwriting our "real" processed flat, let's set the suffix to `_exercise2`.

Hint: `cat flats.lis` to get the name of a flat to use with `showpars`.

[*Solution*]

---

### 1.8.3 Running non-default recipes

A recipe library often contains several recipes. One recipe in the library is set as the default library. If one wants to run a different library, its name can be specified on the command line using the `-r` flag.

As we have seen before to see the list of all available recipes, we can use the command with the `--all` flag.

```
showrecipes ../playdata/example1/N20160102S0270.fits --all

Input file: /Users/klabrie/data/tutorials/niriimg_tutorial/playdata/example1/
↪N20160102S0270.fits
Input tags: {'GEMINI', 'NORTH', 'NIRI', 'UNPREPARED', 'SIDEREAL', 'IMAGE', 'RAW'}
Recipes available for the input file:
   geminidr.niri.recipes.sq.recipes_IMAGE::alignAndStack
   geminidr.niri.recipes.sq.recipes_IMAGE::makeSkyFlat
   geminidr.niri.recipes.sq.recipes_IMAGE::reduce
   geminidr.niri.recipes.qa.recipes_IMAGE::makeSkyFlat
   geminidr.niri.recipes.qa.recipes_IMAGE::reduce
   geminidr.niri.recipes.sq.recipes_IMAGE::alignAndStack
   geminidr.niri.recipes.sq.recipes_IMAGE::makeSkyFlat
   geminidr.niri.recipes.sq.recipes_IMAGE::reduce
```

The strings `sq` and `qa` refer to the reduction mode. The `qa` mode, Quality Assessment, is used internally at Gemini. General users will be using the `sq`, Science Quality, recipes.

**Warning:** The last three "sq" recipes in the list above are really the "ql", quicklook, recipes. This a known bug (circa April 2023). The NIRI quicklook recipes are identical to the science recipes and are just "Python-imported" from the science module, and that import trips the current implementation of `showrecipes`.

The reduction mode is `sq` by default. To change that one uses the `--qa` or `--ql` flags with `reduce`. We will be using the science quality recipes here so we do not need those flags in this version of workshop.

The syntax to select the specific recipe we want to use is:

```
reduce file1.fits file2.fits -r recipename   (optional --ql/--qa)
```

The `-r` option can also be used to primitives as we have seen elsewhere. Just type the name of the primitive instead of the recipe:

```
reduce file1.fits file2.fits -r primitivename
```

Another use of the `-r` option is to run personal recipes rather than the ones distributed with DRAGONS. We will show how that is done in the next chapter.

---

**Exercise - "reduce" 3**

Run the `makeSkyFlat` recipe on the science frames. Set the suffix to `_skyflat`. Note that because the target fills the field-of-view of the science frames, the sky flat in this particular case will not be usable, but it's okay, we are just exploring the `reduce` command-line here.

[*Solution*]

---

### 1.8.4 Overriding the calibration selection

If you wish to force DRAGONS to use a specific processed calibration, overriding the automatic selection, you can use the `--user_cal` flag. Here is a usage example.

```
reduce file1.fits file2.fits --user_cal processed_arc:my_arc.fits
```

---

**Exercise - "reduce" 4**

In the demo, we reduced the flux standard as follow:

```
reduce @stdstar.lis -p darkCorrect:do_cal=skip
```

Modify this command to allow the dark correction to use the processed dark we used for the science frame, `N20160102S0423_dark.fits`.

[*Solution*]

---

### 1.8.5 The "at"-file

We have use the "at"-file already. That `@` symbol followed by the name of the file list, that's an "at"-file. They are very convenient to pass a list of files, but their use is not limited to list of files.

Anything following the `reduce` command can be put in a file and prefixed with the `@` symbol on the `reduce` command line.

Let say that we have a very customized call to `reduce`:

```
reduce @stdstar.lis -r makeSkyFlat --suffix _skyflat \
   -p stackFrames:reject_method=minmax stackFrames:operation=median \
   skyCorrect:scale=False --user_cal processed_dark:N20160102S0423_dark.fits
```

That would be rather tedious to type again and again, or to edit to experiment with the parameter values. Also, when experimenting, it might be nice to keep of record of what is being attempted.

All this can be written to a file and passed to `reduce` with `@`.

Let's write that file, let's call it `params`, any name will do:

```
# I can add comments

-r makeSkyFlat   # in-line comments too
--suffix _skyflat

-p
   stackFrames:reject_method=minmax
   stackFrames:operation=median
   #stackFrames:operation=mean   # I can comment out options to try something else

   skyCorrect:scale=False

--user_cal processed_dark:N20160102S0423_dark.fits

# the file is completely free-form
```

Then to use that file:

```
reduce @stdstar.lis @params
```

The "at"-files can even be nested, for example I could add `@stdstar.lis` to my `params` file:

```
@stdstar.lis

-r makeSkyFlat
--suffix skyflat

-p
   stackFrames:reject_method=minmax
   stackFrames:operation=median

   skyCorrect:scale=False

--user_cal processed_dark=N20160102S0423_dark.fits
```

And just call `@params`, the `@stdstar.lis` will be expanded as if it had been on the command line.

```
reduce @params
```

## 1.9 Customize Recipes

Sometimes the DRAGONS recipes are not quite what is needed. It is possible to customize a recipe and have DRAGONS run it. The easiest way to customize a recipe is to find the most appropriate one, copy it over and edit it.

Here we will show you how to write intermediate results to disk for inspection.

The first step is to find a recipe and copy it to the local directory. Let's find the default NIRI science imaging recipe.

```
showrecipes ../playdata/example1/N20160102S0270.fits
```

The recipe name and the recipe library location returned will look like this:

```
...
Input recipe: reduce
...
Recipe location: /Users/klabrie/condaenvs/public3.10_3.1.0/lib/python3.10/site-packages/
↪geminidr/niri/recipes/sq/recipes_IMAGE.py
...
```

**Note:** The file might end with `.pyc` instead of `.py`. This is the compiled file, the code file is in the same location and named `.py`. That is the file we will copy over. It is not clear at this time why the `.pyc` is sometimes returned instead of the `.py` filename.

```
cp /Users/klabrie/condaenvs/public3.10_3.1.0/lib/python3.10/site-packages/geminidr/niri/
↪recipes/sq/recipes_IMAGE.py .
```

You should now have a file named `recipes_IMAGE.py` in your current directory. Let's rename it for clarity:

```
mv recipes_IMAGE.py myNIRIrecipes.py
```

Use your favorite editor to open `myNIRIrecipes.py` to edit the recipe named `reduce`. We will save some intermediate results: the flat corrected files and the sky corrected frames before alignment and stacking. To do that we need to add `writeOutputs` after `flatCorrect` and `skyCorrect`.

```
p.prepare()
p.addDQ()
p.removeFirstFrame()
p.ADUToElectrons()
p.addVAR(read_noise=True, poisson_noise=True)
p.nonlinearityCorrect()
p.darkCorrect()
p.flatCorrect()
p.writeOutputs()
p.separateSky()
p.associateSky(stream='sky')
p.skyCorrect(instream='sky', mask_objects=False, outstream='skysub')
p.detectSources(stream='skysub')
p.transferAttribute(stream='sky', source='skysub', attribute='OBJMASK')
p.clearStream(stream='skysub')
p.associateSky()
p.skyCorrect(mask_objects=True)
p.writeOutputs()
p.detectSources()
p.adjustWCSToReference()
p.resampleToCommonFrame()
p.stackFrames()
p.writeOutputs()
return
```

Note that some of the primitives in the recipe have parameters set in the recipe itself. Parameter values set in a recipe will always take precedence. Those cannot be changed by the user from the command line.

Now that we have an edited recipe that will write the flat corrected and the pre-alignment sky corrected frames, how do we run it? With `-r` of course! But the syntax is just a touch different. We need to specific both the name of our recipe library (`myNIRIrecipes`) and the name of the recipe (`reduce`).

```
reduce @target.lis -r myNIRIrecipes.reduce
```

The new intermediate outputs have the suffixes `_flatCorrected` and `skySubtracted`.

---

**Exercise - Custom Recipe 1**

Edit the master flat recipe to write the pre-normalized flat to disk, and to display the master flat at the end of the recipe.

Hint: `cat flats.lis` to get the name of a flat to use with `showrecipes` to get the location and name of the NIRI flat recipe library.

[*Solution*]

---

## 1.10 Tools

DRAGONS offers some tools to help with the bookkeeping. We have used `dataselect` already to create lists, we used `showpars` and `showrecipes` to check the primitive parameter settings and the recipes.

Here we will explore `dataselect` a bit more, and introduce `showd` and `typewalk`.

Additional information about the tools can be found in the chapter of the .

### 1.10.1 typewalk

The oddly-named `typewalk` tool allows the user the list the `astrodata` tags (formerly "types") of all the files in a directory, and can recurse ("walk") through subdirectories. It can also be used to select data on tags, but we recommend using the much more flexible `dataselect` for that.

To see the type of data files that we have in `playdata/example1` and see which tags are available for selection:

```
typewalk --dir ../playdata/example1

directory:  /Users/klabrie/data/tutorials/niriimg_tutorial/playdata/example1
 N20160102S0270.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0271.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0272.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0273.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0274.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0275.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0276.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0277.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0278.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0279.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0295.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0296.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0297.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0298.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0299.fits ..... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0363.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
```

(continues on next page)

(continued from previous page)

```
N20160102S0364.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0365.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0366.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0367.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0368.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0369.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0370.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0371.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0372.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0373.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0374.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0375.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0376.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0377.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0378.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0379.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0380.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0381.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0382.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0423.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0424.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0425.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
```

(continues on next page)

(continued from previous page)

```
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0426.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0427.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0428.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0429.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0430.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0431.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0432.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0463.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0464.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0465.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0466.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0467.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0468.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0469.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0470.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0471.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0472.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
```

From that output we see the darks with their `DARK` tag, the flats with the `FLAT` tags, some with `LAMPON`, some with `LAMPOFF`. The science frames do not have the `CAL` (for calibration) tag.

### 1.10.2 showd

`typewalk` shows tags, `showd` shows the values of astrodata . The list available descriptor is included in an appendix of the .

The syntax is:

```
showd -d descriptor_name filenames
showd -d descriptor1,descriptor2,descriptorN filenames
```

The default is to print on the terminal. A comma-separate list can be produced with the `--csv` flag.

**Exercise - Tools 1**

Get the exposure time, filter name, and UT date of all the observations in `playdata/example1`. Use the list from the .

[*Solution*]

### 1.10.3 dataselect

`dataselect` is a powerful tool to create list of exactly the files you want based on tags and descriptors values.

You can include tags, exclude tags, and use Python comparison operators on the descriptors.

Here are a few examples.

Select only darks:

```
dataselect ../playdata/example1/*.fits --tags DARK
```

Select only **non**-darks:

```
dataselect ../playdata/example1/*.fits --xtags DARK
```

Select lamp-on flats:

```
dataselect ../playdata/example1/*.fits --tags FLAT,LAMPON
```

Select any frames with exposure time of 10 seconds:

```
dataselect ../playdata/example1/*.fits --expr="exposure_time==10"
```

Select 20-second darks:

```
dataselect ../playdata/example1/*.fits --tags DARK --expr="exposure_time==20"
```

Select any frames observed on or after 2016-01-03:

```
dataselect ../playdata/example1/*.fits --expr="ut_date>='2016-01-03'"
```

We can combine `dataselect` and `showd`. Here we are going to get the observation type, as defined in the Gemini Observing Tool (OT), for those frames observed on or after 2016-01-03:

```
dataselect ../playdata/example1/*.fits --expr="ut_date>='2016-01-03'" | showd -d␣
↪observation_type
```

The expression in `--expr` is a Python expression using the Python syntax. Note the double == for equality in the examples above. Using a single = is a common mistake.

Also, be careful with the quotes. The external quotes must be different from the internal quotes used around strings.

Finally, the expression can use the `and` and `or` logical operators.

**Exercise - Tools 2**

Knowing that the science frames and the flux standard have an observation type of 'OBJECT', and that the science frames observation class is "science" while the flux standards are "partnerCal", create a list of science frames and a list of flux standards **without** using the object names. Send the output to `showd` to print the object name.

Descriptors: `observation_type`, `observation_class`, `object`.

[*Solution*]

## 1.11 Resources

The DRAGONS documentation is available on "readthedocs" at https://dragons.readthedocs.io/

DRAGONS and all its dependencies are installed with conda. Installation instruction and a lot of information about how to use DRAGONS are provided in the .

The list of is found in .

There are a series of tutorials available on the main DRAGONS portal. They cover reduction of imaging data for each of the current facility imagers.

The code is available on github at: https://github.com/GeminiDRSoftware/DRAGONS

Request for help should be done via the Gemini Helpdesk system using the Topic "DRAGONS".

## 1.12 Solutions to exercises

### 1.12.1 Solutions to Recipe and recipe libraries exercises

#### Solution to Exercise - Recipes 1

The command is used on the file. Highlighted below is the name of the default recipe, `makeProcessedFlat`, and the primitives that will be run.

```
showrecipes ../playdata/example1/N20160102S0373.fits

Recipe not provided, default recipe (makeProcessedFlat) will be used.
Input file: /Users/klabrie/data/tutorials/niriimg_tutorial/playdata/example1/
↪N20160102S0373.fits
Input tags: ['GCALFLAT', 'NORTH', 'AT_ZENITH', 'NON_SIDEREAL', 'AZEL_TARGET', 'RAW',
↪'IMAGE', 'GCAL_IR_ON', 'NIRI', 'GEMINI', 'UNPREPARED', 'LAMPON', 'CAL', 'FLAT']
Input mode: sq
Input recipe: makeProcessedFlat
Matched recipe: geminidr.niri.recipes.sq.recipes_FLAT_IMAGE::makeProcessedFlat
Recipe location: /Users/klabrie/condaenvs/public3.10_3.1.0/lib/python3.10/site-packages/
↪geminidr/niri/recipes/sq/recipes_FLAT_IMAGE.py
Recipe tags: {'FLAT', 'IMAGE', 'CAL', 'NIRI'}
Primitives used:
   p.prepare()
   p.addDQ()
   p.addVAR(read_noise=True)
   p.nonlinearityCorrect()
   p.ADUToElectrons()
   p.addVAR(poisson_noise=True)
   p.makeLampFlat()
   p.normalizeFlat()
   p.thresholdFlatfield()
   p.storeProcessedFlat()
```

### Solution to Exercise - Recipes 2

Again is used. The recipe location, the input file tags, and the recipe tags are highlighted.

```
showrecipes ../playdata/example1/N20160102S0423.fits

Recipe not provided, default recipe (makeProcessedDark) will be used.
Input file: /Users/klabrie/data/tutorials/niriimg_tutorial/playdata/example1/
↪N20160102S0423.fits
Input tags: ['NON_SIDEREAL', 'AT_ZENITH', 'NIRI', 'NORTH', 'UNPREPARED', 'AZEL_TARGET',
↪'GEMINI', 'RAW', 'DARK', 'CAL']
Input mode: sq
Input recipe: makeProcessedDark
Matched recipe: geminidr.niri.recipes.sq.recipes_DARK::makeProcessedDark
Recipe location: /Users/klabrie/condaenvs/public3.10_3.1.0/lib/python3.10/site-packages/
↪geminidr/niri/recipes/sq/recipes_DARK.py
Recipe tags: {'NIRI', 'DARK', 'CAL'}
Primitives used:
   p.prepare()
   p.addDQ(add_illum_mask=False)
   p.addVAR(read_noise=True)
   p.nonlinearityCorrect()
   p.ADUToElectrons()
   p.addVAR(poisson_noise=True)
   p.stackDarks()
   p.storeProcessedDark()
```

### Solution to Exercise - Recipes 3

The option `--all` is used to show all the recipes. Note in the module path the `sq` and the `qa`. The recipe names can be the same but their content will differ depending on the reduction mode selected. Default is always `sq`, science quality.

The last three "sq" recipes are really the "ql" recipes. This a known bug (circa April 2023). The NIRI quicklook recipes are identical to the science recipes and are just "Python imported" from the science module, and that import trips the current implementation of `showrecipes`.

```
showrecipes ../playdata/example1/N20160102S0270.fits --all

geminidr.niri.recipes.sq.recipes_IMAGE::alignAndStack
geminidr.niri.recipes.sq.recipes_IMAGE::makeSkyFlat
geminidr.niri.recipes.sq.recipes_IMAGE::reduce
geminidr.niri.recipes.qa.recipes_IMAGE::makeSkyFlat
geminidr.niri.recipes.qa.recipes_IMAGE::reduce
geminidr.niri.recipes.sq.recipes_IMAGE::alignAndStack
geminidr.niri.recipes.sq.recipes_IMAGE::makeSkyFlat
geminidr.niri.recipes.sq.recipes_IMAGE::reduce
```

To see the content of a specific recipe, name it with the `-r` flag.

```
showrecipes ../playdata/example1/N20160102S0270.fits -r makeSkyFlat
```

Finally, to print the content of the quality assessment, "qa", recipe named `reduce` (also the default in that `qa` library), use the `-m` flag:

```
showrecipes ../playdata/example1/N20160102S0270.fits -r reduce -m qa
```

### 1.12.2 Solutions to Primitives exercises

#### Solution to Exercise - Primitives 1

The first step is get yourself familiarized with the primitive names. This can be done by looking at the recipe.

```
showrecipes ../playdata/example1/N20160102S0270.fits

p.prepare()
p.addDQ()
p.removeFirstFrame()
p.ADUToElectrons()
p.addVAR(read_noise=True, poisson_noise=True)
p.nonlinearityCorrect()
p.darkCorrect()
p.flatCorrect()
p.separateSky()
p.associateSky(stream='sky')
p.skyCorrect(instream='sky', mask_objects=False, outstream='skysub')
p.detectSources(stream='skysub')
p.transferAttribute(stream='sky', source='skysub', attribute='OBJMASK')
p.clearStream(stream='skysub')
p.associateSky()
p.skyCorrect(mask_objects=True)
p.detectSources()
p.adjustWCSToReference()
p.resampleToCommonFrame()
p.stackFrames()
p.writeOutputs()
```

**Question 1:**

This question is about sky correction. The primitive `skyCorrect` is a good bet.

```
showpars ../playdata/example1/N20160102S0270.fits skyCorrect

...
operation            'median'             Averaging operation
Allowed values:
    wtmean   variance-weighted mean
    mean     arithmetic mean
    median   median
    lmedian  low-median
...
```

The default for combining the sky frames is `median`.

**Question 2:**

The second question asks which paramter controls whether or not the dark correction will be run. Let's look at `darkCorrect`.

```
showpars ../playdata/example1/N20160102S0270.fits darkCorrect

Dataset tagged as {'NORTH', 'IMAGE', 'NIRI', 'UNPREPARED', 'SIDEREAL', 'GEMINI', 'RAW'}
Settable parameters on 'darkCorrect':
=====================================
 Name                        Current setting

do_cal                'procmode'           Calibration requirement
Allowed values:
    procmode Use the default rules set by the processingmode.
    force    Require a calibration regardless of theprocessing mode.
    skip         Skip this correction, no calibration required.
    None         Field is optional

suffix                '_darkCorrected'     Filename suffix
dark                  None                 Dark frame
```

The parameter `do_cal` controls the dark correction. Set to "skip".

**Question 3:**

The last question is about the stacking of the reduced frames. At the end of the recipe there's the primitive `stackFrames`.

```
showpars ../playdata/example1/N20160102S0270.fits stackFrames

...
reject_method         'sigclip'            Pixel rejection method
Allowed values:
    none     no rejection
    minmax   reject highest and lowest pixels
    sigclip  reject pixels based on scatter
    varclip  reject pixels based on variance array
...
```

The `reject_method` is the answer. It can be set to `minmax`, to `none`, to `varclip`, or to `sigclip` (currently the default).

### 1.12.3 Solutions to the Explore data exercises

#### Solution to Exercise - Explore 1

**Question 1**

```
reduce -r display N20160102S0271_stack.fits -p frame=2 threshold=None
```

Note that we did not use the `display:` prefix, like in `display:frame=2`. When the primitive name is not specified, the instruction applies to all parameters with that name from any primitives in the recipe. Here, we get away with it because only the `display` primitive is being run.

**Question 2**

```
reduce -r inspect @stdstar.lis -p pause=1
```

### 1.12.4 Solutions to the Local calibration manager exercises

#### Solution to Exercise - Caldb 1

The `dragonsrc` file should look like this:

```
[calibs]
#databases = <where_the_data_package_is>/niriimg_tutorial/playground/cal_manager.db get
databases = <where_the_data_package_is>/niriimg_tutorial/cal_manager.db get
```

`ls <where_the_data_package_is>/niriimg_tutorial` should show a file called `cal_manager.db`. And

```
caldb config

Using configuration files: ('~/.dragons/dragonsrc',)

/Users/klabrie/data/tutorials/niriimg_tutorial/cal_manager.db
  Type:  LocalDB
  Get:   True
  Store: False


Database file: /Users/klabrie/data/tutorials/niriimg_tutorial/cal_manager.db

The calibration dbs are all local, meaning that remote calibrations will not be␣
↪downloaded
```

#### Solution to Exercise - Caldb 2

It is important do to successfully complete *Exercise - Caldb 1* before attempting Exercise 2.

First confirm that the new calibration manager, the one `niriimg_tutorial` is active.

```
caldb config

Using configuration files: ('~/.dragons/dragonsrc',)

/Users/klabrie/data/tutorials/niriimg_tutorial/cal_manager.db
  Type:  LocalDB
  Get:   True
  Store: False


Database file: /Users/klabrie/data/tutorials/niriimg_tutorial/cal_manager.db

The calibration dbs are all local, meaning that remote calibrations will not be␣
↪downloaded
```

**Question 1**

```
caldb add N20160102S0373_flat.fits
```

**Question 2**

```
caldb list

N20160102S0373_flat.fits       /data/workspace/niriimg_tutorial/playground
```

**Question 3**

Edit `dragonsrc`. Comment out the `niriimg_tutorial` path and uncomment the `playground` path.

```
[calibs]
databases = <where_the_data_package_is>/niriimg_tutorial/playground/cal_manager.db get
#databases = <where_the_data_package_is>/niriimg_tutorial/cal_manager.db get
```

Confirm activation with `caldb config`.

```
caldb list

N20160102S0373_flat.fits       /data/workspace/niriimg_tutorial/playground
N20160102S0423_dark.fits       /data/workspace/niriimg_tutorial/playground
```

### 1.12.5 Solutions to the `reduce` exercises

#### Solution to Exercise - "reduce" 1

```
reduce -r detectSources N20160102S0296_stack.fits --suffix _ILoveDRAGONS

Or

reduce -r detectSources N20160102S0296_stack.fits --suffix=_ILoveDRAGONS
```

#### Solution to Exercise - "reduce" 2

```
reduce @flats.lis -p normalizeFlat:scale=mean --suffix _exercise2
```

#### Solution to Exercise - "reduce" 3

```
reduce @target.lis -r makeSkyFlat --suffix _skyflat
```

#### Solution to Exercise - "reduce" 4

While it is not recommended to use a processed dark of the wrong exposure, here is how you would force DRAGONS to use the science's master dark on the flux standard from the demo.

```
reduce @stdstar.lis -p --user_cal processed_dark:N20160102S0423_dark.fits
```

### 1.12.6 Solutions to the Customize recipes exercise

#### Solution to Exercise - Custom Recipe 1

```
showrecipes ../playdata/example1/N20160102S0363.fits
```

```
cp /Users/klabrie/condaenvs/public3.10_3.1.0/lib/python3.10/site-packages/geminidr/niri/
↪recipes/sq/recipes_FLAT_IMAGE.py .
mv recipes_FLAT_IMAGE.py myNIRIflats.py
```

```
p.prepare()
p.addDQ()
p.addVAR(read_noise=True)
p.nonlinearityCorrect()
p.ADUToElectrons()
p.addVAR(poisson_noise=True)
p.makeLampFlat()
p.writeOutputs()
p.normalizeFlat()
p.thresholdFlatfield()
p.storeProcessedFlat()
p.display()
```

```
reduce @flats.lis -r myNIRIflats.makeProcessedFlat
```

### 1.12.7 Solutions to the Tools exercise

#### Solution to Exercise - Tools 1

```
showd -d exposure_time,filter_name,ut_date ../playdata/example1/*.fits

------------------------------------------------------------------------------------
filename                                  exposure_time   filter_name      ut_date
------------------------------------------------------------------------------------
../playdata/example1/N20160102S0270.fits          20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0271.fits          20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0272.fits          20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0273.fits          20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0274.fits          20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0275.fits          20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0276.fits          20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0277.fits          20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0278.fits          20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0279.fits          20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0295.fits          10.005       H_G0203   2016-01-02
../playdata/example1/N20160102S0296.fits          10.005       H_G0203   2016-01-02
../playdata/example1/N20160102S0297.fits          10.005       H_G0203   2016-01-02
../playdata/example1/N20160102S0298.fits          10.005       H_G0203   2016-01-02
../playdata/example1/N20160102S0299.fits          10.005       H_G0203   2016-01-02
../playdata/example1/N20160102S0363.fits          42.001       H_G0203   2016-01-02
```

(continues on next page)

(continued from previous page)

```
../playdata/example1/N20160102S0364.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0365.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0366.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0367.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0368.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0369.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0370.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0371.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0372.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0373.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0374.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0375.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0376.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0377.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0378.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0379.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0380.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0381.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0382.fits          42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0423.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0424.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0425.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0426.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0427.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0428.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0429.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0430.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0431.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0432.fits          20.002         blank   2016-01-02
../playdata/example1/N20160103S0463.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0464.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0465.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0466.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0467.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0468.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0469.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0470.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0471.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0472.fits           1.001         blank   2016-01-03
```

### Solution to Exercise - Tools 2

```
dataselect ../playdata/example1/*.fits --expr='observation_class=="science" and␣
↪observation_type=="OBJECT"' | showd -d object

--------------------------------------------------
filename                                    object
--------------------------------------------------
../playdata/example1/N20160102S0270.fits   SN2014J
../playdata/example1/N20160102S0271.fits   SN2014J
```

(continues on next page)

(continued from previous page)

```
../playdata/example1/N20160102S0272.fits   SN2014J
../playdata/example1/N20160102S0273.fits   SN2014J
../playdata/example1/N20160102S0274.fits   SN2014J
../playdata/example1/N20160102S0275.fits   SN2014J
../playdata/example1/N20160102S0276.fits   SN2014J
../playdata/example1/N20160102S0277.fits   SN2014J
../playdata/example1/N20160102S0278.fits   SN2014J
../playdata/example1/N20160102S0279.fits   SN2014J
```

```
dataselect ../playdata/example1/*.fits --expr='observation_class=="partnerCal" and␣
↪observation_type=="OBJECT"' | showd -d object

-------------------------------------------------
filename                                   object
-------------------------------------------------
../playdata/example1/N20160102S0295.fits    FS 17
../playdata/example1/N20160102S0296.fits    FS 17
../playdata/example1/N20160102S0297.fits    FS 17
../playdata/example1/N20160102S0298.fits    FS 17
../playdata/example1/N20160102S0299.fits    FS 17
```

CHAPTER

TWO

# BASIC DRAGONS WORKSHOP - SHORT VERSION

This is the material for truncated version of the Basic DRAGONS Workshop. The full workshop requires 2 hours. This shorter version is better adapted for a one-hour time slot.

## 2.1 Learning Objectives

This material is built for a 1-hr workshop, planning on 45 minutes for the workshop and 15 minutes for questions and helping participants. This is truncated version of the full workshop.

In this workshop, the participants will gain a very basic understanding of the following topic:

- What is DRAGONS, what it isn't.
- What is a typical DRAGONS workflow.
- What are the principal components of DRAGONS.
- What is a recipe, a recipe library, and a primitive.
- How to do coarse inspection of the data.
- How to use the calibration manager.
- How to customize input parameters and recipes.
- How to control `reduce`.
- What utility tools are available and how to use them.

This workshop tries to cover a lot in a short period of time. The focus will be on some key, often needed elements. The participants should refer to the DRAGONS documentation for more in-depth information.

This workshop uses the command-line interface to DRAGONS. An API is available but will not be covered in this workshop.

### 2.1.1 Setting up

The participants who wish to following along and run the examples and exercises are expected to have already installed DRAGONS on their computer. See the **installation instructions**, .

Follow the Python 3 and DRAGONS instructions, IRAF is not needed. Make sure that you do the steps in the "Configure DRAGONS" section.

**Important: This workshop is written for DRAGONS v3.1**

The participants will need to download the data package to run the examples and the exercises:

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

Download it and unpack it somewhere convenient.

```
cd <somewhere convenient>
tar xvf niriim_tutorial_datapkg-extended-v1.tar
bunzip2 niriimg_tutorial/playdata/example1/*.bz2
```

The datasets are found in the subdirectory `niriimg_tutorial/playdata/example1`, and we will work in the subdirectory named `niriimg_tutorial/playground`.

Finally, for this short version, additional files that would normally be created during the full workshop but are not due to the time constraint will need to be download and added to the `niriimg_tutorial/playground` directory.

https://drive.google.com/file/d/1sJd8bHUFZ3-dNVT3_VllcqplnoSZr4LJ/view?usp=sharing

```
cd <somewhere convenient>/niriimg_tutorial/playground
tar xvzf demo_outputs.tar.gz
```

The content of the `niriimg_tutorial/playground` directory should look like this:

```
N20160102S0271_stack.fits   cal_manager.db                reduce.log
N20160102S0296_stack.fits   calibrations                  stdstar.lis
N20160102S0373_flat.fits    darks20s.lis                 target.lis
N20160102S0423_dark.fits    flats.lis
```

## 2.2 What is DRAGONS

### 2.2.1 What it is

With a little bit of guidance from you, DRAGONS will reduce your Gemini data in a standard way, rapidly, with little fuss. With a bit more guidance from you, DRAGONS will let you optimize the reduction for your specific sources and science objectives.

DRAGONS is operated through the `reduce` command from the shell, or the `Reduce` class from a Python script. You choose. For clarity, we will refer to `reduce` only from now on, but it's the same thing. A call to *reduce* activates the "Recipe System" which is what automates everything.

When `reduce` is called, the first FITS file is opened and identified, then the libraries of algorithms collected as "primitives" and "recipes" will be searched and the best matches will be selected and run on the data.

An important component of the DRAGONS automation infrastructure is the calibration manager. It is operated with the `caldb` command. (Again, there's an API too.) This accesses a lightweight local database that will store information about your locally-processed calibrations. When a primitive is run and needs a calibration, the system will automatically identify the best matched processed calibration and use it, you do not have to specify it on the command line. (Though, you can if you really want to.)

Then there's a series of utilities for sorting through the data and learning about the primitives and the recipes.

All of it uses `astrodata` which is used to open the FITS files. Once a file is opened with `astrodata`, it "knows about itself" and that's how it can find the recipes, primitives, and calibration it needs.

Here's what DRAGONS can look like from a user's point view:

```
dataselect *.fits --tags=FLAT -o flat.lis
dataselect *.fits --expr="object='mycooltarget'" -o sci.lis
caldb init
reduce @flat.lis
caldb add *_flat.fits
reduce @sci.lis
```

The steps are:

1. Create your lists of input data.
2. Initialize your calibration database.
3. Reduce your calibrations and upload the info to the database (the upload can be automated).
4. Reduce your science, it will pick up the calibrations by itself.

As straightforward as a reduction can be, it can be customized to match your needs. The options to the primitives can be adjusted, the recipes themselves can be adjusted.

In this basic introduction to DRAGONS, we will explore all this and learn how to have DRAGONS do our bidding.

### 2.2.2 What it is not

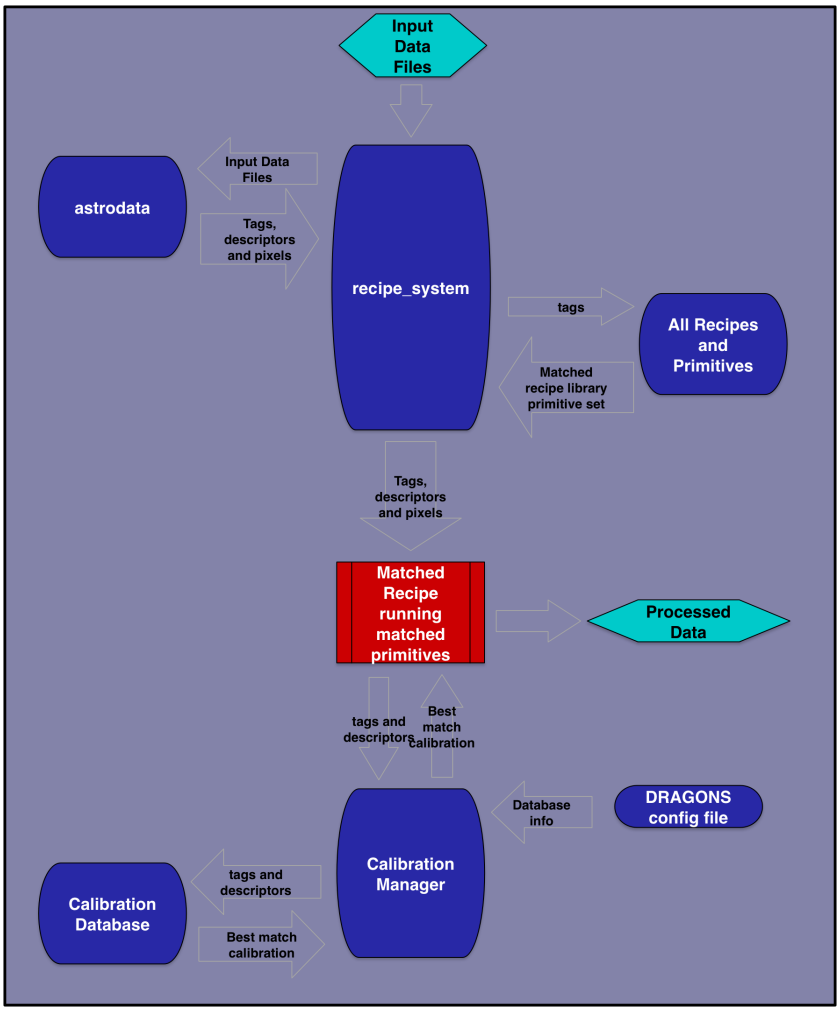
DRAGONS is **not** a data analysis package. DRAGONS prepares the data for analysis but does not offer tools to do the analysis. Also, DRAGONS is also not a replacement for IRAF.

To inspect your data, to do any sort of analysis like photometry, measuring redshift, calculating equivalent width, etc, you will have to find another tool. DRAGONS does provide a primitive to display an image to ds9, but it will just display it. Similarly, there is a tool to plot a 1D spectrum, and again, it will just plot it, it not offer any measurement capabilities.

Any analysis-type tools are beyond the scope of DRAGONS.

### 2.2.3 Components in action

The diagram below illustrates how the components communicate with each other and in particular how the `astrodata` tags and descriptors are passed around to make decisions about the best matched recipe, primitive sets, and calibrations.




## 2.3 What are Recipes, Recipe Libraries, and Primitives

A DRAGONS **recipe** is a set of instructions, called **primitives**, that processes data in a certain way. Depending on the recipe, the input data can be in any state of processing. Most often though, people will want to process Gemini raw data into a master calibration or a processed calibrated output.

A **recipe library** is a collection of recipes. The recipes in a library have one thing in common: *they all apply to the same type of data*. When DRAGONS search for a matching recipe for some input data, it searches for a recipe library. In each library, one recipe is set as the default recipe. To use the others, the user needs to specify the name of the non-default recipes.

A **primitive** is the function that does something to the data. The name of the primitive normally gives you a good idea of what its purpose is.

### 2.3.1 Recipes

This is what a recipe looks like:

```
def reduce(p):
   p.prepare()
   p.addDQ()
   p.removeFirstFrame()
   p.ADUToElectrons()
   p.addVAR(read_noise=True, poisson_noise=True)
   p.nonlinearityCorrect()
   p.darkCorrect()
   p.flatCorrect()
   p.separateSky()
   p.associateSky(stream='sky')
   p.skyCorrect(instream='sky', mask_objects=False, outstream='skysub')
   p.detectSources(stream='skysub')
   p.transferAttribute(stream='sky', source='skysub', attribute='OBJMASK')
   p.clearStream(stream='skysub')
   p.associateSky()
   p.skyCorrect(mask_objects=True)
   p.detectSources()
   p.adjustWCSToReference()
   p.resampleToCommonFrame()
   p.stackFrames()
   p.writeOutputs()
```

A recipe is a Python function that calls "primitives" sequentially. The user does not need to know Python to understand more or less what is being done to the data when the recipe is run. From the recipe above, we can read that the data will be corrected for dark current, for flat field effects, the sky background will be subtracted and the frames will be stacked. The *p* argument is the **primitive set** that matches the input.

### 2.3.2 Recipe libraries

Recipes are stored in recipe libraries, or in Python language, a module. A recipe library can have one or more recipes in them. One recipe is identified as the default recipe.

When DRAGONS searches for a recipe, it is actually searching for a matching recipe library. Once found, it will run the default recipe, unless instructed otherwise by the user.

The recipe libraries are associated with the input files by matching `astrodata` tags (). The tags are qualifiers like "NIRI", "IMAGE", "FLAT". The tags of the first file in the list of inputs are used. Each recipe library is assigned a set of tags that defines which type of data this library is for.

### 2.3.3 Primitives and primitive sets

A primitive is a data reduction step involving a transformation of the data or providing a service. By convention, the primitives are named to convey the scientific meaning of the transformation. For example `biasCorrect` will remove the bias signal from the input data.

A primitive is always a member of a primitive set. It is the primitive set that gets matched to the data by the Recipe System, not the individual primitives.

Technically, a primitive is a method of a primitive class. A primitive class gets associated with the input dataset by matching the `astrodata` tags. Once associated, all the primitives in that class, locally defined or inherited, are available to reduce that dataset. We refer to that collection of primitives as a "primitive set".

### 2.3.4 Primitive input parameters

Also attached to a primitive set are the input parameters for each primitives and the defaults appropriate for that primitive set, that is for that type of data.

For the same generic primitive, the default values for input parameters for NIRI data can be different from the defaults applicable to GMOS data. Even the set of available input parameters can be different, though we try to keep things as uniform as possible.

### 2.3.5 Exploring recipes and primitives

#### showrecipes

allows the user to see which recipe library gets picked up by the system and which recipe is run by default, which others are available.

The syntax is:

```
showrecipes fitsfile_name.fits
```

Let's look at a couple examples. From the `niriimg_tutorial/playdata/example1/` directory:

```
Recipe not provided, default recipe (makeProcessedFlat) will be used.
Input file: /Users/klabrie/data/tutorials/niriimg_tutorial/playdata/example1/
↪N20160102S0373.fits
Input tags: ['GCALFLAT', 'NORTH', 'AT_ZENITH', 'NON_SIDEREAL', 'AZEL_TARGET', 'RAW',
↪'IMAGE', 'GCAL_IR_ON', 'NIRI', 'GEMINI', 'UNPREPARED', 'LAMPON', 'CAL', 'FLAT']
Input mode: sq
Input recipe: makeProcessedFlat
Matched recipe: geminidr.niri.recipes.sq.recipes_FLAT_IMAGE::makeProcessedFlat
Recipe location: /Users/klabrie/condaenvs/public3.10_3.1.0/lib/python3.10/site-packages/
↪geminidr/niri/recipes/sq/recipes_FLAT_IMAGE.py
Recipe tags: {'FLAT', 'IMAGE', 'CAL', 'NIRI'}
Primitives used:
   p.prepare()
   p.addDQ()
   p.addVAR(read_noise=True)
   p.nonlinearityCorrect()
   p.ADUToElectrons()
   p.addVAR(poisson_noise=True)
   p.makeLampFlat()
```

(continues on next page)

(continued from previous page)

```
   p.normalizeFlat()
   p.thresholdFlatfield()
   p.storeProcessedFlat()
```

This contains: the name of the recipe, the location of the recipe library, the :astrodata: tags of the input file, those assigned to the recipe library, and the recipe itself.

A library can have more than one recipe, to see them all use the `--all` flag:

```
showrecipes N20160102S0270.fits --all

Input file: /Users/klabrie/data/tutorials/niriimg_tutorial/playdata/example1/
↪N20160102S0270.fits
Input tags: {'NIRI', 'NORTH', 'GEMINI', 'UNPREPARED', 'IMAGE', 'RAW', 'SIDEREAL'}
Recipes available for the input file:
   geminidr.niri.recipes.sq.recipes_IMAGE::alignAndStack
   geminidr.niri.recipes.sq.recipes_IMAGE::makeSkyFlat
   geminidr.niri.recipes.sq.recipes_IMAGE::reduce
   geminidr.niri.recipes.qa.recipes_IMAGE::makeSkyFlat
   geminidr.niri.recipes.qa.recipes_IMAGE::reduce
   geminidr.niri.recipes.sq.recipes_IMAGE::alignAndStack
   geminidr.niri.recipes.sq.recipes_IMAGE::makeSkyFlat
   geminidr.niri.recipes.sq.recipes_IMAGE::reduce
```

The recipe library for science quality has three recipes: `alignAndStack`, `makeSkyFlat`, and `reduce`.

---

**Note:** Regarding the "sq" and "qa" in the paths. DRAGONS has the concept of "reduction mode". Right now, there are two modes: the science quality mode, "sq", and the quality assessment mode, "qa". You can safely ignore the "qa" mode, it is used exclusively at the observatory, at night, to help with the assessment of the sky conditions and the resulting quality of the data. Everything defaults to "sq".

The last three "sq" recipes are really the "ql" recipes. This a known bug (circa April 2023). The NIRI quicklook recipes are identical to the science recipes and are just "Python imported" from the science module, and that import trips the current implementation of `showrecipes`.

---

To see what a specific recipe looks like, not just the default recipe, use the `-r` flag:

```
showrecipes N20160102S0270.fits -r makeSkyFlat

Input file: /Users/klabrie/data/tutorials/niriimg_tutorial/playdata/example1/
↪N20160102S0270.fits
Input tags: ['UNPREPARED', 'NORTH', 'NIRI', 'SIDEREAL', 'RAW', 'IMAGE', 'GEMINI']
Input mode: sq
Input recipe: makeSkyFlat
Matched recipe: geminidr.niri.recipes.sq.recipes_IMAGE::makeSkyFlat
Recipe location: /Users/klabrie/condaenvs/gemini3_2.1.x_20200925/lib/python3.6/site-
↪packages/dragons-2.1.1-py3.6-macosx-10.7-x86_64.egg/geminidr/niri/recipes/sq/recipes_
↪IMAGE.py
Recipe tags: {'IMAGE', 'NIRI'}
Primitives used:
   p.prepare()
   p.addDQ()
```

(continues on next page)

(continued from previous page)

```
    p.ADUToElectrons()
    p.addVAR(read_noise=True, poisson_noise=True)
    p.nonlinearityCorrect()
    p.darkCorrect()
    p.stackFrames(operation='median', scale=True, outstream='fastsky')
    p.normalizeFlat(stream='fastsky')
    p.thresholdFlatfield(stream='fastsky')
    p.flatCorrect(flat=p.streams['fastsky'][0], outstream='flattened')
    p.detectSources(stream='flattened')
    p.dilateObjectMask(dilation=10, stream='flattened')
    p.addObjectMaskToDQ(stream='flattened')
    p.writeOutputs(stream='flattened')
    p.transferAttribute(source='flattened', attribute='mask')
    p.stackFrames(operation='mean', scale=True, reject_method="minmax", nlow=0, nhigh=1)
    p.normalizeFlat()
    p.thresholdFlatfield()
    p.storeProcessedFlat(force=True)
```

### showpars

Primitive input parameters can be customized. To see the available input parameters, their default values, and range of allowed values, use . The syntax is:

```
showpars filename.fits primitive_name
```

A dataset is required here because the parameters and their defaults do change depending on the type of data. Defaults between instruments can be different because the data has different characteristics. Also, the primitive might be doing slightly different things (eg. optical vs near-IR).

Let's say that we want to reduce NIRI science data, an example of which is file `N20160102S0270.fits` but we want to customize the sky correction, the dark correction, and the rejection method during the final stack. Here's how one would proceed to check for primitive names and parameters names.

```
showrecipes N20160102S0270.fits

...
    p.prepare()
    p.addDQ()
    p.removeFirstFrame()
    p.ADUToElectrons()
    p.addVAR(read_noise=True, poisson_noise=True)
    p.nonlinearityCorrect()
    p.darkCorrect()
    p.flatCorrect()
    p.separateSky()
    p.associateSky(stream='sky')
    p.skyCorrect(instream='sky', mask_objects=False, outstream='skysub')
    p.detectSources(stream='skysub')
    p.transferAttribute(stream='sky', source='skysub', attribute='OBJMASK')
    p.clearStream(stream='skysub')
    p.associateSky()
    p.skyCorrect(mask_objects=True)
```

(continues on next page)

(continued from previous page)

```
   p.detectSources()
   p.adjustWCSToReference()
   p.resampleToCommonFrame()
   p.stackFrames()
   p.writeOutputs()
```

Here I spot the primtives: `skyCorrect`, `darkCorrect`, and `stackFrames`.

If I wanted to turn off the dark correction...

```
showpars N20160102S0270.fits darkCorrect

Dataset tagged as {'NORTH', 'IMAGE', 'NIRI', 'UNPREPARED', 'SIDEREAL', 'GEMINI', 'RAW'}
Settable parameters on 'darkCorrect':
=======================================
 Name                        Current setting

do_cal               'procmode'           Calibration requirement
Allowed values:
    procmode        Use the default rules set by the processingmode.
    force   Require a calibration regardless of theprocessing mode.
    skip    Skip this correction, no calibration required.
    None    Field is optional

suffix               '_darkCorrected'     Filename suffix
dark                 None                 Dark frame
```

I would set `do_cal` to "skip".

If I wanted to change the `operation` done when combining sky frames to a mean...

```
showpars N20160102S0270.fits skyCorrect

...
operation            'median'             Averaging operation
Allowed values:
    mean    arithmetic mean
    wtmean  variance-weighted mean
    median  median
    lmedian low-median
...
```

I would reset `operation` to `mean`.

If I wanted to change the rejection method when doing the final stack...

```
showpars N20160102S0270.fits stackFrames

...
reject_method        'sigclip'            Pixel rejection method
Allowed values:
    none    no rejection
    minmax  reject highest and lowest pixels
    sigclip reject pixels based on scatter
```

(continues on next page)

(continued from previous page)

```
    varclip reject pixels based on variance array
...
```

I would pick one of the allowed value for `reject_method`.

## 2.4 Demo imaging

To get ourselves oriented and get a feel for how data is processed with DRAGONS, we will run a full reduction on our sample data. The full tutorial to reduce NIRI data can be found at .

Here, we go through the steps with quite a bit less discussion than in the full tutorial as the purpose is simply to provide some context and an idea of a typical data reduction flow to those who have not experience it before.

The observations are of an extended source, a nearby galaxy. The instrument used is NIRI. The science sequence is a series of dithers on target with full offset to sky. We will create a master dark, a master flat, a reduced flux standard, and finally create a calibrated, sky-subtracted stack of the science observations.

Let's run all the steps.

1. Set up the local calibration manager
2. Create the file lists
3. Create the calibration files (dark, flat)
4. Reduce the flux standard
5. Reduce the science observations

```
cd <where_the_data_package_is>/niriimg_tutorial/playground
```

### 2.4.1 Set up the local calibration manager

We will discuss the local calibration manager in a later chapter.

The calibration manager is the first thing to set up when starting on a data reduction project. It provides the automated calibration association.

The file to pay attention to is: `~/.dragons/dragonsrc` (DRAGONS v3.1+)

For this example, that file to contain the following:

```
[calibs]
databases = <where_the_data_package_is>/niriimg_tutorial/playground/cal_manager.db get
```

In this configuration, with the "get" flag, DRAGONS will retrieve the processed calibration automatically from that database. You could add "store" after the "get" to have DRAGONS add the processed calibrations to the database automatically (effectively running `caldb add` for you). Here we will add the calibrations it manually.

About the path for `databases`, we recommend the directory you are using to work on the data. But it can be anywhere.

Once the configuration file is set up, initialize the database. This needs to be done only once per database.

```
caldb init
```

This creates a new file, `cal_manager.db`, in the directory specified by `databases`.

### 2.4.2 Create the file lists

The user has to sort the files. DRAGONS is a "User-in-the-Loop" pipeline. We have some tools to help. We will have a closer look at them later.

We have long darks that match the science, a series of lamp-on and lamp-off flats, a set of on-target dithered observations for a flux standard, and a set of on-target dithered and off-target dithered observations for the science target.

```
dataselect ../playdata/example1/*.fits --tags DARK --expr='exposure_time==20' -o␣
↪darks20s.lis

dataselect ../playdata/example1/*.fits --tags FLAT -o flats.lis

dataselect ../playdata/example1/*.fits --expr='object=="FS 17"' -o stdstar.lis

dataselect ../playdata/example1/*.fits --expr='object=="SN2014J"' -o target.lis
```

### 2.4.3 Add static BPM to calibration database

The static BPMs are downloaded from the archive (but included here in our data package).

```
caldb add ../playdata/example1/bpm_20010317_niri_niri_11_full_1amp.fits
```

### 2.4.4 Create the master dark and the master flat

**Dark**

Create the master dark and add it to the calibration manager.

```
reduce @darks20s.lis
caldb add N20160102S0423_dark.fits
```

The @ character before the name of the input file is the "at-file" syntax. We will look into this later.

**Flat**

Create the master flat from the lamp-on and lamp-off flats and add it to the calibration database.

```
reduce @flats.lis
caldb add N20160102S0373_flat.fits
```

### 2.4.5 Reduce the flux standard

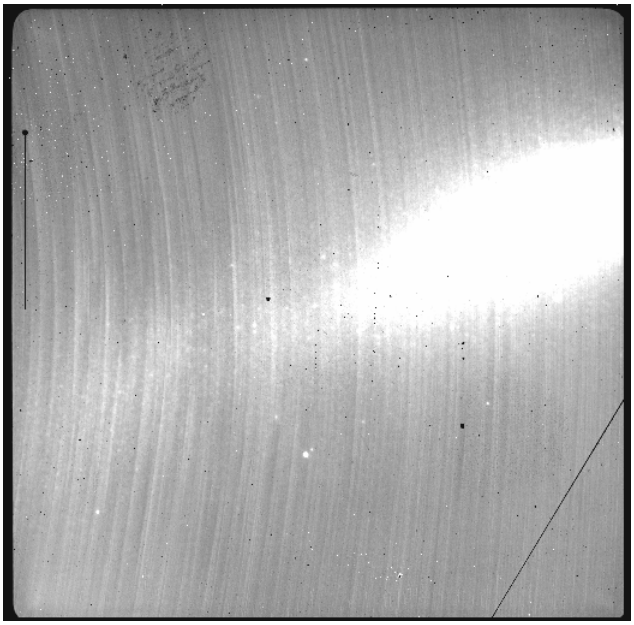
This is short series of on-target dither observations. Darks are not obtained for NIRI flux standard so we turn that step off. The flat will be picked up from the calibration database.

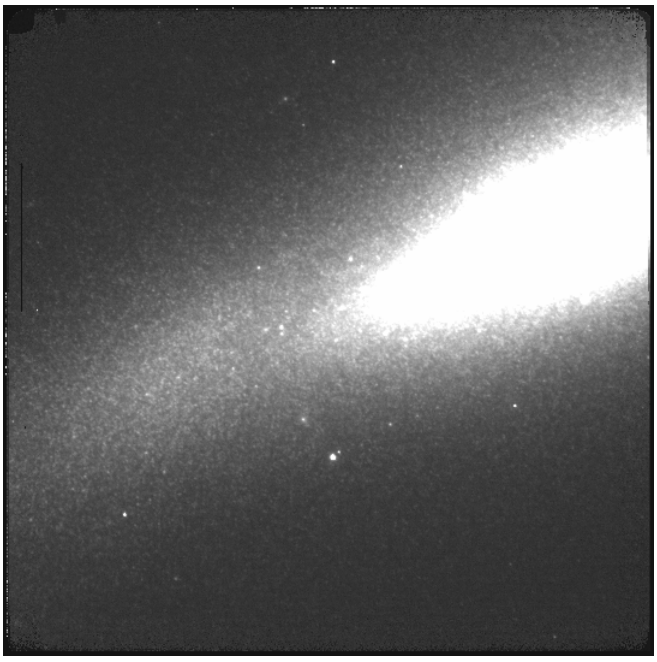
```
reduce @stdstar.lis -p darkCorrect:do_cal=skip
```

### 2.4.6 Reduce the science observations

The sequence is a set of dither on target with full offsets to sky. DRAGONS will sort them during the sky correction. All calibrations are picked up automatically from the calibration database. Because the target fills the field of view, we need to turn off the scaling of the sky frames.

```
reduce @target.lis -p skyCorrect:scale_sky=False
```





## 2.5 Explore data

Once we have reduced data, we probably want to look at. In fact, even before we start the reduction, we should look at the raw data in case of issues.

DRAGONS is **not** an analysis package. But it does offer a basic way to display images.

The `display` and `inspect` primitives can be used to visually inspect the data with `ds9`. Those are primitives not recipes, yet `reduce` can run individual primitives.

---

**Note:** `ds9` must be launched by the user ahead of running the primitive.

---

### 2.5.1 Display

To display an image to `ds9`:

```
reduce -r display N20160102S0271_stack.fits
```

It is possible to display to different buffer in `ds9`, eg. to allow blinking between the frames.

```
reduce -r display N20160102S0271_stack.fits -p frame=2
```

### 2.5.2 Inspect

It is recommended to inspect the raw data prior to reducing the data. A quick visual inspection can catch obviously bad data, or flag some frames for more detailed inspection. Having to run display on each frame individually would be painful. The primitive `inspect` can be used here. It accepts a list of file and will display each frame in sequence, adding a short pause between the frames. The length of the pause can be controlled with the `pause` parameter.

```
reduce -r inspect @stdstar.lis
```

```
reduce -r inspect @stdstar.lis -p pause=1
```

## 2.6 Local calibration manager

The local calibration manager is the tool that will allow DRAGONS to find the most appropriate processed calibration for the data being reduced.

The local calibration manager contains all the same rules as the Gemini Observatory Archive, but works locally, with your files, without internet, and alongside DRAGONS to make the calibration association as seamless as possible. As we have seen in the demo earlier, we create the master calibrations, we add them to the database, and when DRAGONS (`reduce`) needs one, it gets matched and retrieved automatically.

The system uses a lightweight `sqlite` database that will **not** store the data file, it will store only information **about** the file, like its location on disk, and key astrodata tags and descriptors.

> **Warning:** If you move or delete a processed calibration on disk after having entered it in the database, the calibration manager will no longer be able to find it. The database does not contain the file, just information about the file.

See the documentation for more information, including information about the programming interface.

### 2.6.1 Configuration and initialization

**Warning:** The information provided here applies to DRAGONS version 3.1 (and newer). The configuration and features of the calibration manager in the previous versions, 3.0 and older, are substantially different. Please refer to a previous version of this workhop if you are using DRAGONS 3.0 or older,

https://collection-of-dragons-workshops.readthedocs.io/en/v1.0.0/BasicDRAGONS-1hr/sphinx/caldb.html

#### Configuration

The behavior and configuration of the local calibration manager are controlled in the file `~/.dragons/dragonsrc`. When you first install DRAGONS, you will have to create that directory and that file.

To complete the exercises, your `dragonsrc` file must look like this:

```
[calibs]
databases = <where_the_data_package_is>/niriimg_tutorial/playground/cal_manager.db get
```

The value of `databases` can be any **existing** path on your machine. The calibration database will be created, and then expected to be in that directory. The database name can be anything, here we name it `cal_manager.db` for continuity with pre-v3.1 DRAGONS.

We recommend setting up a new database for each observing project. It helps keep things clean and avoids calibrations from other programs being unexpectedly picked up. (If they were picked up, they would be a match, but it might get confusing if you are expecting another calibration to be picked up.)

Here is an example of a `dragonsrc` with several `databases` entries. Note that at all times, **only one** `databases` directive is active, the others are commented out.

```
[calibs]
databases = /Users/klabrie/data/tutorials/niriimg_tutorial/playground/cal_manager.db get
#databases = /Users/klabrie/data/tutorials/gmosimg_tutorial/playground/cal_manager.db get

#databases = /Users/klabrie/data/workspace/GMOS540feature/cal_manager.db get
```

#### Initialization

Once your `dragonsrc` is set up, you can create a **new** database with

```
caldb init
```

If the database already exists, that command will refuse to work. This ensures that you will not wipe the database accidentally. **The `init` is required only once.** If you already have the database created, maybe even populated, you can switch back to it simply by ensuring `databases` points to it.

---

**Exercise - Caldb 1**

Create a new database in the `niri_tutorial` directory. Keep the `playground` entry that should already be there, but deactivate it.

Confirm with `ls` that a new `cal_manager.db` has been created. You can also use `caldb config` to confirm that the new database is active.

[*Solution*]

### 2.6.2 Usage

The usage is fairly basic. One can `list` the content, `add` content, and `remove` content. DRAGONS will retrieve content from the database. If the "store" flag is set, DRAGONS will also automatically add content to the database.

> **Warning:** In DRAGONS version 3.0 and older, DRAGONS cannot add content automatically, the user **must** add the processed calibrations to the database with an explicit `caldb add`.

To verify which database is currently active, use the `config` option:

```
caldb config
```

To see the content of the database:

```
caldb list
```

To add information about a processed calibration to the database:

```
caldb add name_of_file.fits
```

And finally, to delete information about one file from the database:

```
caldb remove name_of_file.fits
```

`remove` will only remove the entry in the database, it will not remove the file on disk.

**Exercise - Caldb 2**

Do *Exercise - Caldb 1* first.

1. Add the flat field from the demo (`N20160102S0373_flat.fits`) to the new calibration manager created in the first exercise.
2. Show the content of that database is indeed just that file.
3. Reactivate the original database (the one in `playground` that we used for the demo) and list its content. Both the dark and the flat should now be listed.

[*Solution*]

## 2.7 Using reduce

The reduce command (and the Reduce class if you are using the programming interface) is what drives DRAGONS. The command has a lot of options. We have used a few already. Let's review those and show a few more.

First, to see the list of options available, use the -h flag:

```
reduce -h
```

### 2.7.1 Setting the suffix of the final outputs

It is not possible to set the name of the final output. This seems odd, but here is why. There might be more than one output, there can be dozens, it depends on the recipe. It is not reasonable to ask the user for a dozen different output strings on the command line, it also interferes with the automation.

We are still exploring solutions to this problem.

In the meantime, the user can set the suffix of the outputs. Each primitive has a meaningful suffix attached to it, for example _sourcesDetected for the primitive detectSources. The output of a reduce call will get the suffix assigned to the last primitive of the recipe.

If you wish to override that, you can set the --suffix option.

```
reduce @targets --suffix=_mytest
```

### 2.7.2 Customizing parameters

We have seen earlier how to check the available input parameters and their default settings for a given frame and primitive. The command does that. For example:

```
showpars ../playdata/example1/N20160102S0363.fits normalizeFlat
```

The syntax to change the value of an input parameter is as follow:

```
reduce file.fits -p primitive:parameter=value
```

---

**Exercise - "reduce" 1**

White down the reduce command to reduce the flats from the demo (flats.lis) but this time set the scaling for normalizeFlat flat to mean. To avoid overwriting our "real" processed flat, let's set the suffix to _exercise1.

Hint #1: cat flats.lis to get the name of a flat to use with showpars.

Hint #2: See the *command used in the demo*

[*Solution*]

---

### 2.7.3 Running non-default recipes

A recipe library often contains several recipes. One recipe in the library is set as the default library. If one wants to run a different library, its name can be specified on the command line using the `-r` flag.

As we have seen before to see the list of all available recipes, we can use the command with the `--all` flag.

```
showrecipes ../playdata/example1/N20160102S0270.fits --all

Input file: /Users/klabrie/data/tutorials/niriimg_tutorial/playdata/example1/
↪N20160102S0270.fits
Input tags: {'GEMINI', 'NORTH', 'NIRI', 'UNPREPARED', 'SIDEREAL', 'IMAGE', 'RAW'}
Recipes available for the input file:
   geminidr.niri.recipes.sq.recipes_IMAGE::alignAndStack
   geminidr.niri.recipes.sq.recipes_IMAGE::makeSkyFlat
   geminidr.niri.recipes.sq.recipes_IMAGE::reduce
   geminidr.niri.recipes.qa.recipes_IMAGE::makeSkyFlat
   geminidr.niri.recipes.qa.recipes_IMAGE::reduce
   geminidr.niri.recipes.sq.recipes_IMAGE::alignAndStack
   geminidr.niri.recipes.sq.recipes_IMAGE::makeSkyFlat
   geminidr.niri.recipes.sq.recipes_IMAGE::reduce
```

The strings `sq` and `qa` refer to the reduction mode. The `qa` mode, Quality Assessment, is used internally at Gemini. General users will be using the `sq`, Science Quality, recipes.

> **Warning:** The last three "sq" recipes in the list above are really the "ql", Quicklook, recipes. This a known bug (circa April 2023). The NIRI quicklook recipes are identical to the science recipes and are just "Python-imported" from the science module, and that import trips the current implementation of `showrecipes`.

The reduction mode is `sq` by default. To change that, one uses the `--qa` or soon `--ql` flags with `reduce`. We will be using the science quality recipes here so we do not need those flags in this version of workshop.

The syntax to select the specific recipe we want to use is:

```
reduce file1.fits file2.fits -r recipename   (optional --ql/--qa)
```

The `-r` option can also be used to primitives as we have seen elsewhere. Just type the name of the primitive instead of the recipe:

```
reduce file1.fits file2.fits -r primitivename
```

Another use of the `-r` option is to run personal recipes rather than the ones distributed with DRAGONS. We will show how that is done in the next chapter.

---

**Exercise - "reduce" 2**

Write the command to use the `makeSkyFlat` recipe on the science frames. Set the suffix to `_skyflat`.

Note that because the target fills the field-of-view of the science frames, the sky flat in this particular case will not be usable, but it's okay, we are just exploring the `reduce` command-line here.

[*Solution*]

---

### 2.7.4 Overriding the calibration selection

If you wish to force DRAGONS to use a specific processed calibration, overriding the automatic selection, you can use the `--user_cal` flag. Here is a usage example.

```
reduce file1.fits file2.fits --user_cal processed_<cal>:my_cal.fits
```

Allowed values for "<cal>": dark, bias, flat, arc, standard. Eg:

```
reduce @mylist.lis --user_cal processed_arc:my_special_arc.fits
```

---

**Exercise - "reduce" 3**

In the demo, we reduced the flux standard as follow:

```
reduce @stdstar.lis -p darkCorrect:do_cal=skip
```

Modify this command to allow the dark correction to use the processed dark we used for the science frame, `N20160102S0423_dark.fits`.

[*Solution*]

---

## 2.8 Customize Recipes

Sometimes the DRAGONS recipes are not quite what is needed. It is possible to customize a recipe and have DRAGONS run it. The easiest way to customize a recipe is to find the most appropriate one, copy it over and edit it.

Here we will show you how to write intermediate results to disk for inspection.

The first step is to find a recipe and copy it to the local directory. Let's find the default NIRI science imaging recipe.

```
showrecipes ../playdata/example1/N20160102S0270.fits
```

The recipe name and the recipe library location returned will look like this:

```
...
Input recipe: reduce
...
Recipe location: /Users/klabrie/condaenvs/public3.10_3.1.0/lib/python3.10/site-packages/
↪geminidr/niri/recipes/sq/recipes_IMAGE.py
...
```

---

**Note:** The file might end with `.pyc` instead of `.py`. This is the compiled file, the code file is in the same location and named `.py`. That is the file we will copy over. It is not clear at this time why the `.pyc` is sometimes returned instead of the `.py` filename.

---

```
cp /Users/klabrie/condaenvs/public3.10_3.1.0/lib/python3.10/site-packages/geminidr/niri/
↪recipes/sq/recipes_IMAGE.py .
```

You should now have a file named `recipes_IMAGE.py` in your current directory. Let's rename it for clarity:

```
mv recipes_IMAGE.py myNIRIrecipes.py
```

Use your favorite editor to open `myNIRIrecipes.py` to edit the recipe named `reduce`. We will save some intermediate results: the flat corrected files and the sky corrected frames before alignment and stacking. To do that we need to add `writeOutputs` after `flatCorrect` and `skyCorrect`.

```
p.prepare()
p.addDQ()
p.removeFirstFrame()
p.ADUToElectrons()
p.addVAR(read_noise=True, poisson_noise=True)
p.nonlinearityCorrect()
p.darkCorrect()
p.flatCorrect()
p.writeOutputs()
p.separateSky()
p.associateSky(stream='sky')
p.skyCorrect(instream='sky', mask_objects=False, outstream='skysub')
p.detectSources(stream='skysub')
p.transferAttribute(stream='sky', source='skysub', attribute='OBJMASK')
p.clearStream(stream='skysub')
p.associateSky()
p.skyCorrect(mask_objects=True)
p.writeOutputs()
p.detectSources()
p.adjustWCSToReference()
p.resampleToCommonFrame()
p.stackFrames()
p.writeOutputs()
return
```

Note that some of the primitives in the recipe have parameters set in the recipe itself. Parameter values set in a recipe will always take precedence. Those cannot be changed by the user from the command line.

Now that we have an edited recipe that will write the flat corrected and the pre-alignment sky corrected frames, how do we run it? With `-r` of course! But the syntax is just a touch different. We need to specific both the name of our recipe library (`myNIRIrecipes`) and the name of the recipe (`reduce`).

```
reduce @target.lis -r myNIRIrecipes.reduce
```

The new intermediate outputs have the suffixes `_flatCorrected` and `_skySubtracted`.

---

**Exercise - Custom Recipe 1**

Edit the master flat recipe to write the pre-normalized flat to disk, and to display the master flat at the end of the recipe.

Hint: `cat flats.lis` to get the name of a flat to use with `showrecipes` to get the location and name of the NIRI flat recipe library.

[*Solution*]

---

## 2.9 Tools

DRAGONS offers some tools to help with the bookkeeping. We have used `dataselect` already to create lists, we used `showpars` and `showrecipes` to check the primitive parameter settings and the recipes.

Here we will explore `dataselect` a bit more, and introduce `showd` and `typewalk`.

Additional information about the tools can be found in the chapter of the .

### 2.9.1 typewalk

The oddly-named `typewalk` tool allows the user the list the `astrodata` tags (formerly "types") of all the files in a directory, and can recurse ("walk") through subdirectories. It can also be used to select data on tags, but we recommend using the much more flexible `dataselect` for that.

To see the type of data files that we have in `playdata/example1` and see which tags are available for selection:

```
typewalk --dir ../playdata

directory:  /Users/klabrie/data/tutorials/niriimg_tutorial/playdata
 N20160102S0270.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0271.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0272.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0273.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0274.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0275.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0276.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0277.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0278.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0279.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0295.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0296.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0297.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0298.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0299.fits ............... (GEMINI) (IMAGE) (NIRI) (NORTH) (RAW) (SIDEREAL)␣
↪(UNPREPARED)
 N20160102S0363.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
```

(continues on next page)

(continued from previous page)

```
N20160102S0364.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0365.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0366.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0367.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0368.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0369.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0370.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0371.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0372.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_OFF) (GEMINI) (IMAGE) (LAMPOFF) (NIRI) (NON_SIDEREAL) (NORTH) (RAW)␣
↪(UNPREPARED)
N20160102S0373.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0374.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0375.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0376.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0377.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0378.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0379.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0380.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0381.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0382.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (FLAT) (GCALFLAT)␣
↪(GCAL_IR_ON) (GEMINI) (IMAGE) (LAMPON) (NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0423.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0424.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
N20160102S0425.fits ..... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
```

(continues on next page)

(continued from previous page)

```
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0426.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0427.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0428.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0429.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0430.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0431.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160102S0432.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0463.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0464.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0465.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0466.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0467.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0468.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0469.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0470.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0471.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
 N20160103S0472.fits ............... (AT_ZENITH) (AZEL_TARGET) (CAL) (DARK) (GEMINI)␣
↪(NIRI) (NON_SIDEREAL) (NORTH) (RAW) (UNPREPARED)
```

From that output we see the darks with their `DARK` tag, the flats with the `FLAT` tags, some with `LAMPON`, some with `LAMPOFF`. The science frames do not have the `CAL` (for calibration) tag.

### 2.9.2 showd

`typewalk` shows tags, `showd` shows the values of astrodata . The list available descriptor is included in an appendix of the .

The syntax is:

```
showd -d descriptor_name filenames
showd -d descriptor1,descriptor2,descriptorN filenames
```

The default is to print on the terminal. A comma-separate list can be produced with the `--csv` flag.

**Exercise - Tools 1**

Get the exposure time, filter name, and UT date of all the observations in `playdata/example1`. Use the list from the
.

[*Solution*]

### 2.9.3 dataselect

`dataselect` is a powerful tool to create list of exactly the files you want based on tags and descriptors values.

You can include tags, exclude tags, and use Python comparison operators on the descriptors.

Here are a few examples.

Select only darks:

```
dataselect ../playdata/example1/*.fits --tags DARK
```

Select only darks and send the results to a file:

```
dataselect ../playdata/example1/*.fits --tags DARK -o darks.lis

cat darks.lis

# Includes tags: ['DARK']
# Excludes tags: []
# Descriptor expression: None
../playdata/example1/N20160102S0423.fits
../playdata/example1/N20160102S0424.fits
...
../playdata/example1/N20160103S0471.fits
../playdata/example1/N20160103S0472.fits
```

Select only **non**-darks:

```
dataselect ../playdata/example1/*.fits --xtags DARK
```

Select lamp-on flats:

```
dataselect ../playdata/example1/*.fits --tags FLAT,LAMPON
```

Select any frames with exposure time of 10 seconds:

```
dataselect ../playdata/example1/*.fits --expr="exposure_time==10"
```

Select 20-second darks:

```
dataselect ../playdata/example1/*.fits --tags DARK --expr="exposure_time==20"
```

Select any frames observed on or after 2016-01-03:

```
dataselect ../playdata/example1/*.fits --expr="ut_date>='2016-01-03'"
```

We can combine `dataselect` and `showd`. Here we are going to get the observation type, as defined in the Gemini Observing Tool (OT), for those frames observed on or after 2016-01-03:

```
dataselect ../playdata/example1/*.fits --expr="ut_date>='2016-01-03'" | showd -d␣
↪observation_type
```

The expression in `--expr` is a Python expression using the Python syntax. Note the double == for equality in the examples above. Using a single = is a common mistake.

Also, be careful with the quotes. The external quotes must be different from the internal quotes used around strings.

Finally, the expression can use the `and` and `or` logical operators.

---

**Exercise - Tools 2**

Knowing that the science frames and the flux standard have an observation type of 'OBJECT', and that the science frames observation class is "science" while the flux standards are "partnerCal", create a list of science frames and a list of flux standards **without** using the object names. Send the output to `showd` to print the object name.

Descriptors: `observation_type`, `observation_class`, `object`.

[*Solution*]

---

## 2.10 Resources

The DRAGONS documentation is available on "readthedocs" at https://dragons.readthedocs.io/

DRAGONS and all its dependencies are installed with conda. Installation instruction and a lot of information about how to use DRAGONS are provided in the .

The list of is found in .

There are a series of tutorials available on the main DRAGONS portal. They cover reduction of imaging data for each of the current facility imagers.

The code is available on github at: https://github.com/GeminiDRSoftware/DRAGONS

Request for help should be done via the Gemini Helpdesk system using the Topic "DRAGONS".

## 2.11 Solutions to exercises

### 2.11.1 Solutions to the Local calibration manager exercises

#### Solution to Exercise - Caldb 1

The `dragonrc` file should look like this:

```
[calibs]
#databases = <where_the_data_package_is>/niriimg_tutorial/playground/cal_manager.db get
databases = <where_the_data_package_is>/niriimg_tutorial/cal_manager.db get
```

`ls <where_the_data_package_is>/niriimg_tutorial` should show a file called `cal_manager.db`. And

```
caldb config

Using configuration files: ('~/.dragons/dragonsrc',)

/Users/klabrie/data/tutorials/niriimg_tutorial/cal_manager.db
  Type:  LocalDB
  Get:   True
  Store: False


Database file: /Users/klabrie/data/tutorials/niriimg_tutorial/cal_manager.db

The calibration dbs are all local, meaning that remote calibrations will not be␣
↪downloaded
```

### Solution to Exercise - Caldb 2

It is important do to successfully complete *Exercise - Caldb 1* before attempting Exercise 2.

First confirm that the new calibration manager, the one `niriimg_tutorial` is active.

```
caldb config

Using configuration files: ('~/.dragons/dragonsrc',)

/Users/klabrie/data/tutorials/niriimg_tutorial/cal_manager.db
  Type:  LocalDB
  Get:   True
  Store: False


Database file: /Users/klabrie/data/tutorials/niriimg_tutorial/cal_manager.db

The calibration dbs are all local, meaning that remote calibrations will not be␣
↪downloaded
```

**Question 1**

```
caldb add N20160102S0373_flat.fits
```

**Question 2**

```
caldb list

N20160102S0373_flat.fits       /data/workspace/niriimg_tutorial/playground
```

**Question 3**

Edit `dragonsrc`. Comment out the `niriimg_tutorial` path and uncomment the `playground` path.

```
[calibs]
databases = <where_the_data_package_is>/niriimg_tutorial/playground/cal_manager.db get
#databases = <where_the_data_package_is>/niriimg_tutorial/cal_manager.db get
```

Confirm activation with `caldb config`.

```
caldb list

N20160102S0373_flat.fits       /data/workspace/niriimg_tutorial/playground
N20160102S0423_dark.fits       /data/workspace/niriimg_tutorial/playground
```

### 2.11.2 Solutions to the `reduce` exercises

#### Solution to Exercise - "reduce" 1

```
reduce @flats.lis -p normalizeFlat:scale=mean --suffix _exercise1
```

#### Solution to Exercise - "reduce" 2

```
reduce @target.lis -r makeSkyFlat --suffix _skyflat
```

#### Solution to Exercise - "reduce" 3

While it is not recommended to use a processed dark of the wrong exposure, here is how you would force DRAGONS to use the science's master dark on the flux standard from the demo.

```
reduce @stdstar.lis --user_cal processed_dark:N20160102S0423_dark.fits
```

### 2.11.3 Solutions to the Customize recipes exercise

#### Solution to Exercise - Custom Recipe 1

```
showrecipes ../playdata/example1/N20160102S0363.fits
```

```
cp /Users/klabrie/condaenvs/public3.10_3.1.0/lib/python3.10/site-packages/geminidr/niri/
↪recipes/sq/recipes_FLAT_IMAGE.py .
mv recipes_FLAT_IMAGE.py myNIRIflats.py
```

```
def makeProcessedFlat(p):

    p.prepare()
    p.addDQ()
    p.addVAR(read_noise=True)
    p.nonlinearityCorrect()
    p.ADUToElectrons()
    p.addVAR(poisson_noise=True)
    p.makeLampFlat()
    p.writeOutputs()
    p.normalizeFlat()
    p.thresholdFlatfield()
```

(continues on next page)

(continued from previous page)

```
    p.storeProcessedFlat()
    p.display()
```

```
reduce @flats.lis -r myNIRIflats.makeProcessedFlat
```

### 2.11.4 Solutions to the Tools exercise

#### Solution to Exercise - Tools 1

```
showd -d exposure_time,filter_name,ut_date ../playdata/example1/*.fits

--------------------------------------------------------------------------------
filename                                exposure_time   filter_name      ut_date
--------------------------------------------------------------------------------
../playdata/example1/N20160102S0270.fits         20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0271.fits         20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0272.fits         20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0273.fits         20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0274.fits         20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0275.fits         20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0276.fits         20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0277.fits         20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0278.fits         20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0279.fits         20.002       H_G0203   2016-01-02
../playdata/example1/N20160102S0295.fits         10.005       H_G0203   2016-01-02
../playdata/example1/N20160102S0296.fits         10.005       H_G0203   2016-01-02
../playdata/example1/N20160102S0297.fits         10.005       H_G0203   2016-01-02
../playdata/example1/N20160102S0298.fits         10.005       H_G0203   2016-01-02
../playdata/example1/N20160102S0299.fits         10.005       H_G0203   2016-01-02
../playdata/example1/N20160102S0363.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0364.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0365.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0366.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0367.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0368.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0369.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0370.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0371.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0372.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0373.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0374.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0375.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0376.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0377.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0378.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0379.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0380.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0381.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0382.fits         42.001       H_G0203   2016-01-02
../playdata/example1/N20160102S0423.fits         20.002         blank   2016-01-02
```

(continues on next page)

(continued from previous page)

```
../playdata/example1/N20160102S0424.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0425.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0426.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0427.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0428.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0429.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0430.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0431.fits          20.002         blank   2016-01-02
../playdata/example1/N20160102S0432.fits          20.002         blank   2016-01-02
../playdata/example1/N20160103S0463.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0464.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0465.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0466.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0467.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0468.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0469.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0470.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0471.fits           1.001         blank   2016-01-03
../playdata/example1/N20160103S0472.fits           1.001         blank   2016-01-03
```

### Solution to Exercise - Tools 2

```
dataselect ../playdata/example1/*.fits --expr='observation_class=="science" and␣
↪observation_type=="OBJECT"' | showd -d object

-------------------------------------------------
filename                                   object
-------------------------------------------------
../playdata/example1/N20160102S0270.fits   SN2014J
../playdata/example1/N20160102S0271.fits   SN2014J
../playdata/example1/N20160102S0272.fits   SN2014J
../playdata/example1/N20160102S0273.fits   SN2014J
../playdata/example1/N20160102S0274.fits   SN2014J
../playdata/example1/N20160102S0275.fits   SN2014J
../playdata/example1/N20160102S0276.fits   SN2014J
../playdata/example1/N20160102S0277.fits   SN2014J
../playdata/example1/N20160102S0278.fits   SN2014J
../playdata/example1/N20160102S0279.fits   SN2014J
```

```
dataselect ../playdata/example1/*.fits --expr='observation_class=="partnerCal" and␣
↪observation_type=="OBJECT"' | showd -d object

-------------------------------------------------
filename                                   object
-------------------------------------------------
../playdata/example1/N20160102S0295.fits    FS 17
../playdata/example1/N20160102S0296.fits    FS 17
../playdata/example1/N20160102S0297.fits    FS 17
../playdata/example1/N20160102S0298.fits    FS 17
../playdata/example1/N20160102S0299.fits    FS 17
```

- *Basic DRAGONS Workshop* - Standard Version (~2hrs)
- *Basic DRAGONS Workshop - Short Version* - Short Version (~1hr)

CHAPTER

THREE

# INDICES AND TABLES

- genindex
- modindex
- search