Tutorial Series - GMOS Longslit Data Reduction with DRAGONS Documentation

Release 3.0.2

Kathleen Labrie

July 2022

CONTENTS:

Document ID

PIPE-USER-121-_GMOSLS-DRTutorial

CHAPTER

ONE

OVERVIEW

This is a tutorial for the reduction of GMOS longslit spectroscopic data with DRAGONS.

> **Warning:** DRAGONS v3.0.0 is **NOT** approved for science quality reduction of GMOS Longslit data. This version of DRAGONS should be used on GMOS Longslit data only for quicklook reduction and inspection. Please continue to use the Gemini IRAF package to produce your science quality products for GMOS Longslit while we work on providing that service on DRAGONS in a future release.

GMOS is an imager and spectrograph offering longslit spectroscopy, multi-object spectroscopy (MOS), and integral field spectroscopy. This tutorial focuses on the longslit spectroscopy. For a tutorial on the reduction of GMOS imaging data, see GMOS Imaging Data Reduction Tutorial.

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

The GMOS longslit tutorial series for now contains one scientific example, the reduction of an observation of a single stellar source with dither in both wavelength and spatial direction.

The reduction can be done in two different ways:

- From the terminal using the command line.
- From Python using the DRAGONS classes and functions.

We show how to run the same reduction using both methods.

- **Dithered point source**
  - *Example 1-A: Dithered Point Source Longslit - Using the "reduce" command line*
  - *Example 1-B: Dithered Point Source Longslit - Using the "Reduce" class*

More examples will be added in the future.

See the to install the software if you have not already.

CHAPTER

TWO

SETTING UP AND TUTORIAL DATASETS

2.1 Downloading the tutorial datasets

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

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

Download it and unpack it somewhere convenient.

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

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

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

2.2 Datasets descriptions

2.2.1 Dataset for Example 1: Dithered Point Source Longslit

This is a GMOS longslit observation. The primary target is a DB white dwarf candidate. We will use this observation to show how a basic longslit sequence is reduced with DRAGONS. The sequence dithers along the dispersion axis and along the slit. DRAGONS will adjust for the difference in central wavelength and spatial positions, and then stack the aligned spectra automatically.

The data uses the B600 grating on GMOS South. The central wavelengths are 515 nm and 530 nm. We are using a subset of the original sequence to keep the data volume low. The effective sequence is:

```
(Science - Flat - Science - Arc) - (Science - Flat - Science - Arc)
```

with the first group of four at 515 and the second at 530 nm. The spectrophotometry standard was obtained about a month before the science observation.

The calibrations we use for this example are:

- Biases. The science and the standard observations are often taken with different Region-of-Interest (ROI) as the standard uses only the central area. Therefore we need two sets of biases, one for the science's "Full Frame" ROI, and one for the standard's "Central Spectrum" ROI.

- Spectroscopic flats taken with each of the science and standard observations.
- Arcs, for both the science and the standard observations.
- A spectrophotometric standard.

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

Science	S20171022S0087,89 (515 nm) S20171022S0095,97 (530 nm)
Science biases	S20171021S0265-269 S20171023S0032-036
Science flats	S20171022S0088 (515 nm) S20171022S0096 (530 nm)
Science arcs	S20171022S0092 (515 nm) S20171022S0099 (530 nm)
Standard (LTT2415)	S20170826S0160 (515 nm)
Standard biases	S20170825S0347-351 S20170826S0224-228
Standard flats	S20170826S0161 (515 nm)
Standard arc	S20170826S0162 (515 nm)

CHAPTER

THREE

EXAMPLE 1-A: DITHERED POINT SOURCE LONGSLIT - USING THE "REDUCE" COMMAND LINE

In this example we will reduce a GMOS longslit observation of a DB white dwarf candidate using the "" command that is operated directly from the unix shell. Just open a terminal and load the DRAGONS conda environment to get started.

This observation dithers along the slit and along the dispersion axis.

3.1 The dataset

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

The dataset specific to this example is described in:

> *Dataset for Example 1: Dithered Point Source Longslit*.

Here is a copy of the table for quick reference.

Science	S20171022S0087,89 (515 nm) S20171022S0095,97 (530 nm)
Science biases	S20171021S0265-269 S20171023S0032-036
Science flats	S20171022S0088 (515 nm) S20171022S0096 (530 nm)
Science arcs	S20171022S0092 (515 nm) S20171022S0099 (530 nm)
Standard (LTT2415)	S20170826S0160 (515 nm)
Standard biases	S20170825S0347-351 S20170826S0224-228
Standard flats	S20170826S0161 (515 nm)
Standard arc	S20170826S0162 (515 nm)

3.2 Set up the Local Calibration Manager

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

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

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

```
[calibs]
standalone = True
database_dir = <where_the_data_package_is>/gmosls_tutorial/playground
```

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

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

Then initialize the calibration database:

```
caldb init
```

That's it. It is ready to use.

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

3.3 Create file lists

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

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

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

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

```
cd <path>/gmosls_tutorial/playground
```

3.3.1 Two lists for the biases

The science observations and the spectrophotometric standard observations were obtained using different regions-of-interest (ROI). So we will need two master biases, one "Full Frame" for the science and one "Central Spectrum" for the standard.

We can use to select biases for each ROIs.

Given the data that we have in the `playdata` directory, we can create our GMOS-S bias list using the tags and expression using the ROI settings. Remember, this will always depend on what you have in your raw data directory. For easier selection criteria, you might want to keep raw data from different programs in different directories.

First, let's see which biases we have for in our raw data directory.

```
dataselect ../playdata/*.fits --tags BIAS | showd -d detector_roi_setting

------------------------------------------------------
filename                         detector_roi_setting
------------------------------------------------------
../playdata/S20170825S0347.fits       Central Spectrum
../playdata/S20170825S0348.fits       Central Spectrum
../playdata/S20170825S0349.fits       Central Spectrum
../playdata/S20170825S0350.fits       Central Spectrum
../playdata/S20170825S0351.fits       Central Spectrum
../playdata/S20170826S0224.fits       Central Spectrum
../playdata/S20170826S0225.fits       Central Spectrum
```

(continues on next page)

(continued from previous page)

```
../playdata/S20170826S0226.fits       Central Spectrum
../playdata/S20170826S0227.fits       Central Spectrum
../playdata/S20170826S0228.fits       Central Spectrum
../playdata/S20171021S0265.fits             Full Frame
../playdata/S20171021S0266.fits             Full Frame
../playdata/S20171021S0267.fits             Full Frame
../playdata/S20171021S0268.fits             Full Frame
../playdata/S20171021S0269.fits             Full Frame
../playdata/S20171023S0032.fits             Full Frame
../playdata/S20171023S0033.fits             Full Frame
../playdata/S20171023S0034.fits             Full Frame
../playdata/S20171023S0035.fits             Full Frame
../playdata/S20171023S0036.fits             Full Frame
```

We can see the two groups that differ on their ROI.

```
dataselect ../playdata/*.fits --tags BIAS --expr='detector_roi_setting=="Central Spectrum
↪"' -o biasesstd.lis
dataselect ../playdata/*.fits --tags BIAS --expr='detector_roi_setting=="Full Frame"' -o␣
↪biasessci.lis
```

3.3.2 A list for the flats

The GMOS longslit flats are not normally stacked. The default recipe does not stack the flats. This allows us to use only one list of the flats. Each will be reduced individually, never interacting with the others.

If you have multiple programs and you want to reduce only the flats for that program, you might want to use the `program_id` descriptor

Or, like here, we have only one set of flats, so we will just gather them all together.

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

3.3.3 A list for the arcs

The GMOS longslit arcs are not normally stacked. The default recipe does not stack the arcs. This allows us to use only one list of arcs. Each will be reduce individually, never interacting with the others.

The arcs normally share the `program_id` with the science observations if you find that you need more accurate sorting. We do not need it here.

```
dataselect ../playdata/*.fits --tags ARC -o arcs.lis
```

3.3.4 A list for the spectrophotometric standard star

If a spectrophotometric standard is recognized as such by DRAGONS, it will receive the Astrodata tag `STANDARD`. All spectrophotometric standards normally used at Gemini are in the DRAGONS list of recognized standards.

```
dataselect ../playdata/*.fits --tags STANDARD -o std.lis
```

3.3.5 A list for the science observation

The science observations are what is left, anything that is not a calibration or assigned the tag `CAL`.

If we had multiple targets, we would need to split them into separate list. To inspect what we have we can use and together.

```
dataselect ../playdata/*.fits --xtags CAL | showd -d object

------------------------------------------
filename                            object
------------------------------------------
../playdata/S20171022S0087.fits   J2145+0031
../playdata/S20171022S0089.fits   J2145+0031
../playdata/S20171022S0095.fits   J2145+0031
../playdata/S20171022S0097.fits   J2145+0031
```

Here we only have one object from the same sequence. We would not need any expression, just excluding calibrations would be sufficient. But we demonstrate here how one would specify the object name for a more surgical selection.

```
dataselect ../playdata/*.fits --xtags CAL --expr='object=="J2145+0031"' -o sci.lis
```

3.4 Master Bias

We create the master biases with the “” command, then add them to the local calibration manager with the command.

```
reduce @biasesstd.lis
reduce @biasessci.lis
caldb add *_bias.fits
```

The master biases are `S20170825S0347_bias.fits` and `S20171021S0265_bias.fits`; this information is in both the terminal log and the log file. The `@` character before the name of the input file is the “at-file” syntax. More details can be found in the documentation.

Note: The file name of the output processed bias is the file name of the first file in the list with `_bias` appended as a suffix. This the general naming scheme used by “”.

3.5 Master Flat Field

GMOS longslit flat field are normally obtained at night along with the observation sequence to match the telescope and instrument flexure. The matching flat nearest in time to the target observation is used to flat field the target. The central wavelength, filter, grating, binning, gain, and read speed must match.

Because of the flexure, GMOS longslit flat field are not stacked. Each is reduced and used individually. The default recipe takes that into account.

We can send all the flats, regardless of characteristics, to and each will be reduce individually. When a calibration is needed, in this case, a master bias, the best match will be obtained automatically from the local calibration manager.

```
reduce @flats.lis --ql
caldb add *_flat.fits
```

We can bulk-add the master flats to the local calibration manager as shown above.

Note: GMOS longslit reduction is currently available only for quicklook reduction. The science quality recipes do not exist, hence the use of the `--ql` flag to activate the "quicklook" recipes.

3.6 Processed Arc - Wavelength Solution

GMOS longslit arc can be obtained at night with the observation sequence, if requested by the program, but are often obtained at the end of the night or the following afternoon instead. Like the spectroscopic flats, they are not stacked which means that they can be sent to reduce all together and will be reduced individually.

The wavelength solution is automatically calculated and has been found to be quite reliable. There might be cases where it fails; inspect the `*_mosaic.pdf` plot and the RMS of `determineWavelengthSolution` in the logs to confirm a good solution.

```
reduce @arcs.lis --ql
caldb add *_arc.fits
```

Note: Failures of the wavelength solution calculation are not easy to fix in quicklook mode. It might be better to simply not use the arc at all and rely on the approximate solution instead. When the science quality package is released, there will be interactive tools to fix a bad solution. Remember, this version only offers quicklook reduction for GMOS longslit.

3.7 Processed Standard - Sensitivity Function

The GMOS longslit spectrophotometric standards are normally taken when there is a hole in the queue schedule, often when the weather is not good enough for science observations. One standard per configuration, per program is the norm. If you dither along the dispersion axis, most likely only one of the positions will have been used for the spectrophotometric standard. This is normal for baseline calibrations at Gemini. The standard is used to calculate the sensitivity function. It has been shown that a difference of 10 or so nanometers in central wavelength setting does not significantly impact the spectrophotometric calibration.

The reduction of the standard will be using a master bias, a master flat, and a processed arc. If those have been added to the local calibration manager, they will be picked up automatically.

```
reduce @std.lis --ql
caldb add *_standard.fits
```

To inspect the spectrum:

```
dgsplot S20170826S0160_ql_standard.fits 1
```

To learn how to plot a 1-D spectrum with matplotlib using the WCS from a Python script, see Tips and Tricks *Plot a 1-D spectrum*.

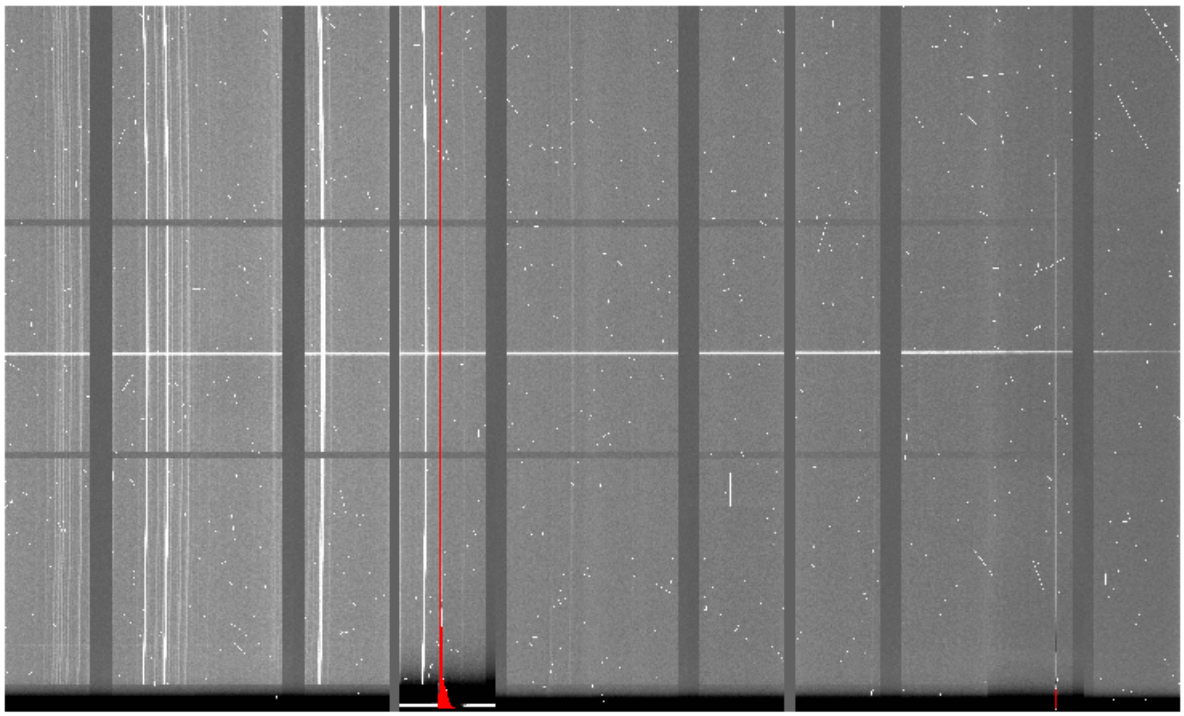
The sensitivity function is stored within the processed standard spectrum. To learn how to plot it, see Tips and Tricks *Inspect the sensitivity function*.

3.8 Science Observations

The science target is a DB white dwarf candidate. The sequence has four images that were dithered spatially and along the dispersion axis. DRAGONS will register the four images in both directions, align and stack them before extracting the 1-D spectrum.

Note: In this observation, there is only one source to extract. If there were multiple sources in slits, regardless of whether they are of interest to the program, DRAGONS will locate them, trace them, and extract them automatically. Each extracted spectrum is stored in an individual extension in the output multi-extension FITS file.

This is what one raw image looks like.



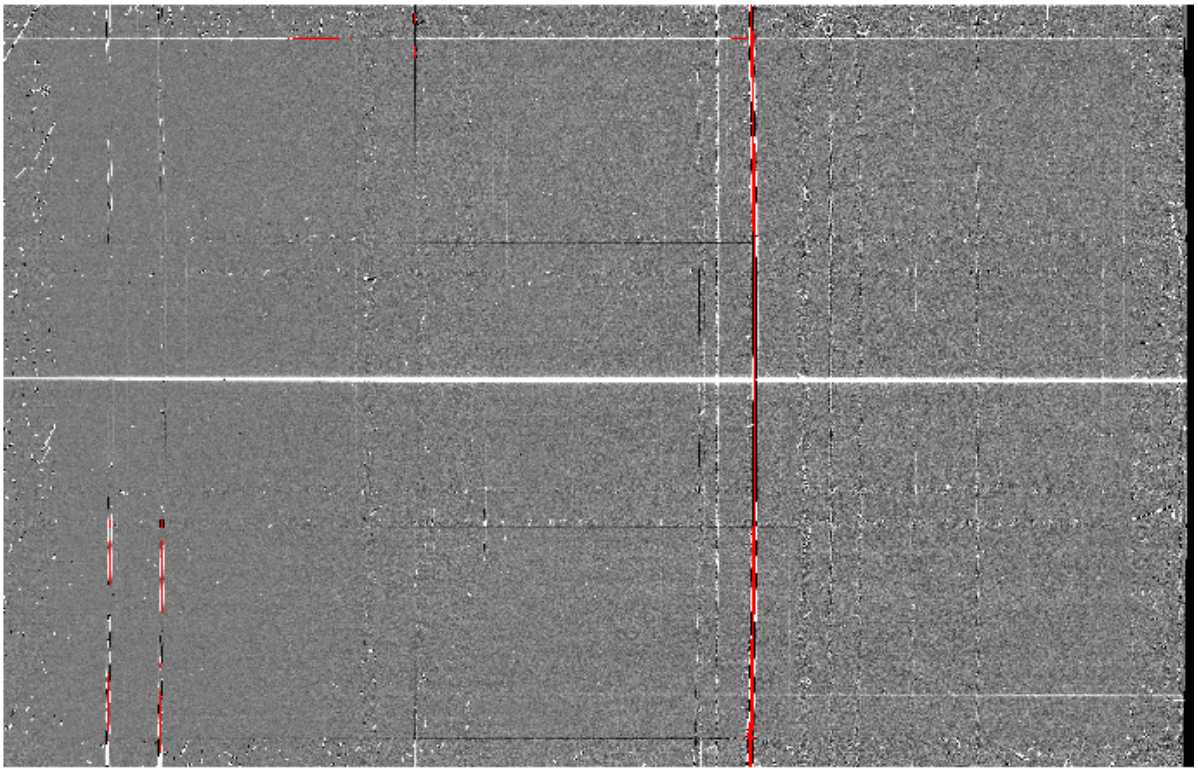
With the master bias, the master flat, the processed arcs (one for each of the grating position, aka central wavelength), and the processed standard in the local calibration manager, to reduce the science observations and extract the 1-D spectrum, one only needs to do as follows.

```
reduce @sci.lis --ql
```

This produces a 2-D spectrum (`S20171022S0087_2D.fits`) which has been bias corrected, flat fielded, QE-corrected, wavelength-calibrated, corrected for distortion, sky-subtracted, and stacked. It also produces the 1-D spectrum (`S20171022S0087_1D.fits`) extracted from that 2-D spectrum. The 1-D spectrum is flux calibrated with the sensitivity function from the spectrophotometric standard. The 1-D spectra are stored as 1-D FITS images in extensions of the output Multi-Extension FITS file.

This is what the 2-D spectrum looks like.

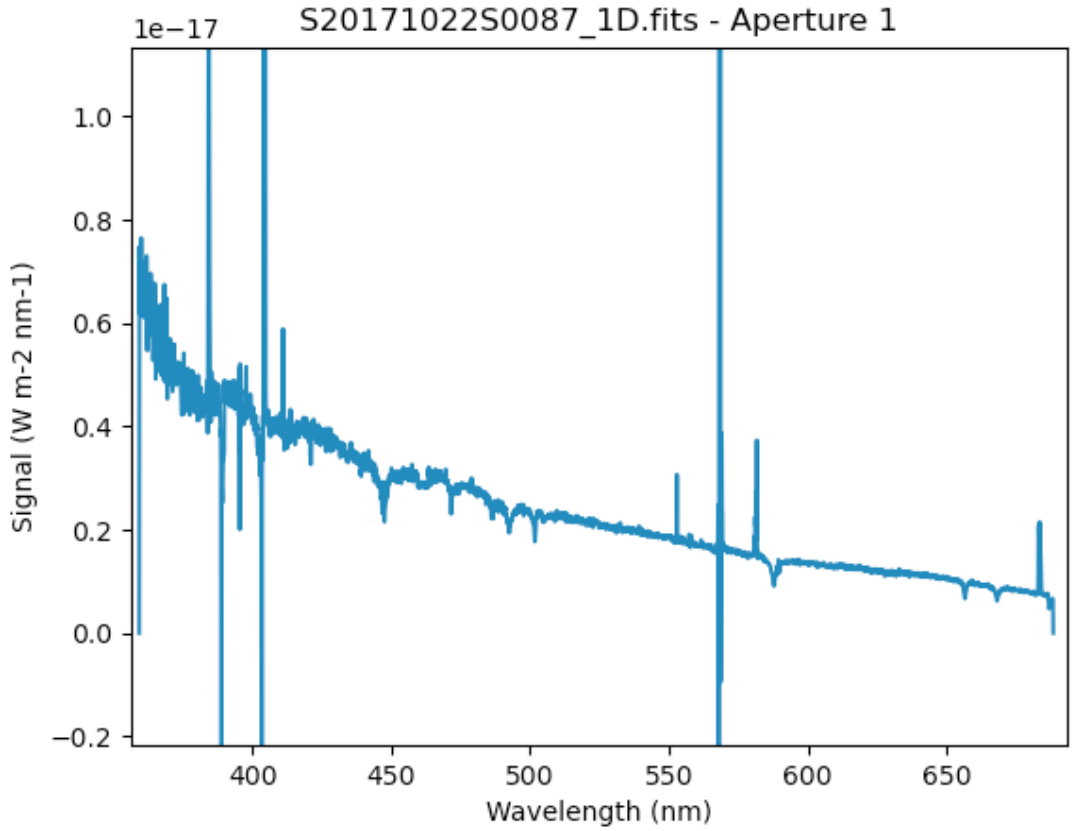
```
reduce -r display S20171022S0087_2D.fits
```



The apertures found are listed in the log for the `findApertures` just before the call to `traceApertures`. Information about the apertures are also available in the header of each extracted spectrum: `XTRACTED`, `XTRACTLO`, `XTRACTHI`, for aperture center, lower limit, and upper limit, respectively.

This is what the 1-D flux-calibrated spectrum of our sole target looks like.

```
dgsplot S20171022S0087_1D.fits 1
```



To learn how to plot a 1-D spectrum with matplotlib using the WCS from a Python script, see Tips and Tricks *Plot a 1-D spectrum*.

If you need an ascii representation of the spectum, you can use the primitive `write1DSpectra` to extra the values from the FITS file.

```
reduce -r write1DSpectra S20171022S0087_1D.fits
```

The primitive outputs in the various formats offered by `astropy.Table`. To see the list, use .

```
showpars S20171022S0087_1D.fits write1DSpectra
```

To use a different format, set the `format` parameters.

```
reduce -r write1DSpectra -p format=ascii.ecsv extension='ecsv' S20171022S0087_1D.fits
```

CHAPTER

FOUR

EXAMPLE 1-B: DITHERED POINT SOURCE LONGSLIT - USING THE "REDUCE" CLASS

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

4.1 The dataset

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

The dataset specific to this example is described in:

> *Dataset for Example 1: Dithered Point Source Longslit*.

Here is a copy of the table for quick reference.

Science	S20171022S0087,89 (515 nm) S20171022S0095,97 (530 nm)
Science biases	S20171021S0265-269 S20171023S0032-036
Science flats	S20171022S0088 (515 nm) S20171022S0096 (530 nm)
Science arcs	S20171022S0092 (515 nm) S20171022S0099 (530 nm)
Standard (LTT2415)	S20170826S0160 (515 nm)
Standard biases	S20170825S0347-351 S20170826S0224-228
Standard flats	S20170826S0161 (515 nm)
Standard arc	S20170826S0162 (515 nm)

4.2 Setting up

First, navigate to your work directory in the unpacked data package.

```
cd <path>/gmosls_tutorial/playground
```

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

4.2.1 Importing libraries

```
import glob

import astrodata
import gemini_instruments
from recipe_system.reduction.coreReduce import Reduce
from recipe_system import cal_service
from gempy.adlibrary import dataselect
```

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

4.2.2 Setting up the logger

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

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

4.2.3 Set up the Local Calibration Manager

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

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

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

```
[calibs]
standalone = True
database_dir = <where_the_data_package_is>/gmosls_tutorial/playground
```

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

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

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

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

cal_service.set_calservice()
```

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

4.3 Create file lists

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

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

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

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

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

We show several usage examples below.

4.3.1 Two lists for the biases

We have two sets for biases: one for the science observation, one for the spectrophotometric standard observation. The science observations and the spectrophotometric standard observations were obtained using different regions-of-interest (ROI). So we will need two master biases, one "Full Frame" for the science and one "Central Spectrum" for the standard.

To inspect data for specific , and to figure out how to build our expression, we can loop through the biases and print the value for the descriptor of interest, here `detector_roi_setting`.

```
17 all_biases = dataselect.select_data(all_files, ['BIAS'])
18 for bias in all_biases:
19     ad = astrodata.open(bias)
20     print(bias, '  ', ad.detector_roi_setting())
```

```
../playdata/S20170825S0347.fits    Central Spectrum
../playdata/S20170825S0348.fits    Central Spectrum
../playdata/S20170825S0349.fits    Central Spectrum
../playdata/S20170825S0350.fits    Central Spectrum
../playdata/S20170825S0351.fits    Central Spectrum
../playdata/S20170826S0224.fits    Central Spectrum
../playdata/S20170826S0225.fits    Central Spectrum
../playdata/S20170826S0226.fits    Central Spectrum
../playdata/S20170826S0227.fits    Central Spectrum
../playdata/S20170826S0228.fits    Central Spectrum
../playdata/S20171021S0265.fits    Full Frame
../playdata/S20171021S0266.fits    Full Frame
../playdata/S20171021S0267.fits    Full Frame
../playdata/S20171021S0268.fits    Full Frame
../playdata/S20171021S0269.fits    Full Frame
../playdata/S20171023S0032.fits    Full Frame
../playdata/S20171023S0033.fits    Full Frame
../playdata/S20171023S0034.fits    Full Frame
../playdata/S20171023S0035.fits    Full Frame
../playdata/S20171023S0036.fits    Full Frame
```

We can clearly see the two groups of biases above. Let's split them into two lists.

```
biasstd = dataselect.select_data(
    all_files,
    ['BIAS'],
    [],
    dataselect.expr_parser('detector_roi_setting=="Central Spectrum"')
)

biassci = dataselect.select_data(
    all_files,
    ['BIAS'],
    [],
    dataselect.expr_parser('detector_roi_setting=="Full Frame"')
)
```

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

4.3.2 A list for the flats

The GMOS longslit flats are not normally stacked. The default recipe does not stack the flats. This allows us to use only one list of the flats. Each will be reduced individually, never interacting with the others.

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

4.3.3 A list for the arcs

The GMOS longslit arcs are not normally stacked. The default recipe does not stack the arcs. This allows us to use only one list of arcs. Each will be reduce individually, never interacting with the others.

```
arcs = dataselect.select_data(all_files, ['ARC'])
```

4.3.4 A list for the spectrophotometric standard star

If a spectrophotometric standard is recognized as such by DRAGONS, it will receive the Astrodata tag `STANDARD`. To be recognized, the name of the star must be in a lookup table. All spectrophotometric standards normally used at Gemini are in that table.

```
stdstar = dataselect.select_data(all_files, ['STANDARD'])
```

4.3.5 A list for the science observation

The science observations are what is left, anything that is not a calibration or assigned the tag `CAL`.

First, let's have a look at the list of objects.

```
37 all_science = dataselect.select_data(all_files, [], ['CAL'])
38 for sci in all_science:
39     ad = astrodata.open(sci)
40     print(sci, '  ', ad.object())
```

On line 37, remember that the second argument contains the tags to **include** (`tags`) and the third argument is the list of tags to **exclude** (`xtags`).

```
../playdata/S20171022S0087.fits    J2145+0031
../playdata/S20171022S0089.fits    J2145+0031
../playdata/S20171022S0095.fits    J2145+0031
../playdata/S20171022S0097.fits    J2145+0031
```

In this case we only have one target. If we had more than one, we would need several lists and we could use the `object` descriptor in an expression. We will do that here to show how it would be done. To be clear, the `dataselect.expr_parser` argument is not necessary in this specific case.

```
41 scitarget = dataselect.select_data(
42     all_files,
43     [],
44     ['CAL'],
45     dataselect.expr_parser('object=="J2145+0031"')
46 )
```

4.4 Master Bias

We create the master biases with the `Reduce` class. We will run it twice, once of each of the two raw bias lists, then add the master biases produced to the local calibration manager with the `caldb` instance. The output is written to disk and its name is stored in the `Reduce` instance. The calibration service expects the name of a file on disk.

```
47 reduce_biasstd = Reduce()
48 reduce_biassci = Reduce()
49 reduce_biasstd.files.extend(biasstd)
50 reduce_biassci.files.extend(biassci)
51 reduce_biasstd.runr()
52 reduce_biassci.runr()
53
54 caldb.add_cal(reduce_biasstd.output_filenames[0])
55 caldb.add_cal(reduce_biassci.output_filenames[0])
```

The two master biases are: `S20170825S0347_bias.fits` and `S20171021S0265_bias.fits`.

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

4.5 Master Flat Field

GMOS longslit flat fields are normally obtained at night along with the observation sequence to match the telescope and instrument flexure. The matching flat nearest in time to the target observation is used to flat field the target. The central wavelength, filter, grating, binning, gain, and read speed must match.

Because of the flexure, GMOS longslit flat field are not stacked. Each is reduced and used individually. The default recipe takes that into account.

We can send all the flats, regardless of characteristics, to `Reduce` and each will be reduce individually. When a calibration is needed, in this case, a master bias, the best match will be obtained automatically from the local calibration manager.

```
reduce_flats = Reduce()
reduce_flats.files.extend(flats)
reduce_flats.mode = 'ql'
reduce_flats.runr()

for f in reduce_flats.output_filenames:
    caldb.add_cal(f)
```

Note: GMOS longslit reduction is currently available only for quicklook reduction. The science quality recipes do not exist, hence the use of the `ql` mode to activate the "quicklook" recipes.

4.6 Processed Arc - Wavelength Solution

GMOS longslit arc can be obtained at night with the observation sequence, if requested by the program, but are often obtained at the end of the night instead. In this example, the arcs have been obtained at night, as part of the sequence.

Like the spectroscopic flats, they are not stacked which means that they can be sent to reduce all to together and will be reduced individually.

The wavelength solution is automatically calculated and the algorithm has been found to be quite reliable. There might be cases where it fails; inspect the `*_mosaic.pdf` plot and the RMS of `determineWavelengthSolution` in the logs to confirm a good solution.

```
reduce_arcs = Reduce()
reduce_arcs.files.extend(arcs)
reduce_arcs.mode = 'ql'
reduce_arcs.runr()

for f in reduce_arcs.output_filenames:
    caldb.add_cal(f)
```

Note: Failures of the wavelength solution calculation are not easy to fix in quicklook mode. It might be better to simply not use the arc at all and rely on the approximate solution instead. When the science quality package is released, there will be interactive tools to fix a bad solution. Remember, this version only offers quicklook reduction for GMOS longslit.

4.7 Processed Standard - Sensitivity Function

The GMOS longslit spectrophotometric standards are normally taken when there is a hole in the queue schedule, often when the weather is not good enough for science observations. One standard per configuration, per program is the norm. If you dither along the dispersion axis, most likely only one of the positions will have been used for the spectrophotometric standard. This is normal for baseline calibrations at Gemini. The standard is used to calculate the sensitivity function. It has been shown that a difference of 10 or so nanometers does not significantly impact the spectrophotometric calibration.

The reduction of the standard will be using a master bias, a master flat, and a processed arc. If those have been added to the local calibration manager, they will be picked up automatically.

```
reduce_std = Reduce()
reduce_std.files.extend(stdstar)
reduce_std.mode = 'ql'
reduce_std.runr()

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

To inspect the spectrum:

```
from gempy.adlibrary import plotting
import matplotlib.pyplot as plt

ad = astrodata.open(reduce_std.output_filenames[0])
plt.ioff()
plotting.dgsplot_matplotlib(ad, 1)
plt.ion()
```

To learn how to plot a 1-D spectrum with matplotlib using the WCS from a Python script, see Tips and Tricks *Plot a 1-D spectrum*.

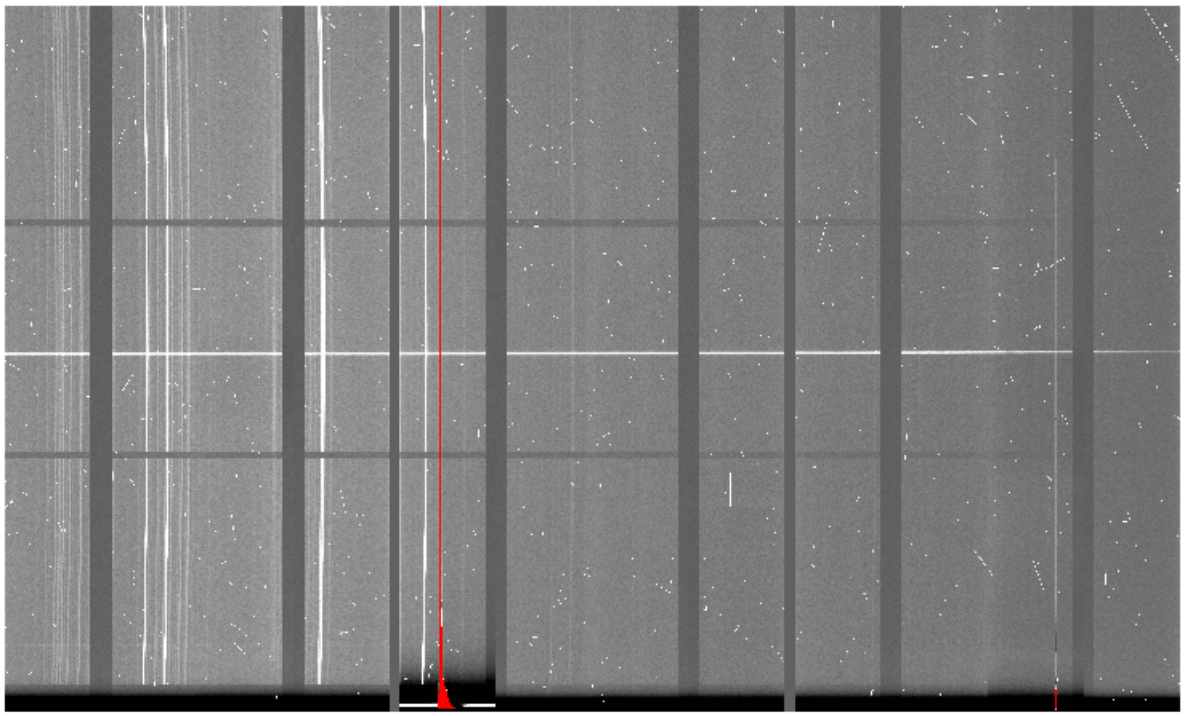
The sensitivity function is stored within the processed standard spectrum. To learn how to plot it, see Tips and Tricks *Inspect the sensitivity function*.

4.8 Science Observations

The science target is a DB white dwarf candidate. The sequence has four images that were dithered spatially and along the dispersion axis. DRAGONS will register the four images in both directions, align and stack them before extracting the 1-D spectrum.

Note: In this observation, there is only one source to extract. If there were multiple sources in slits, regardless of whether they are of interest to the program, DRAGONS will locate them, trace them, and extract them automatically. Each extracted spectrum is stored in an individual extension in the output multi-extension FITS file.

This is what one raw image looks like.



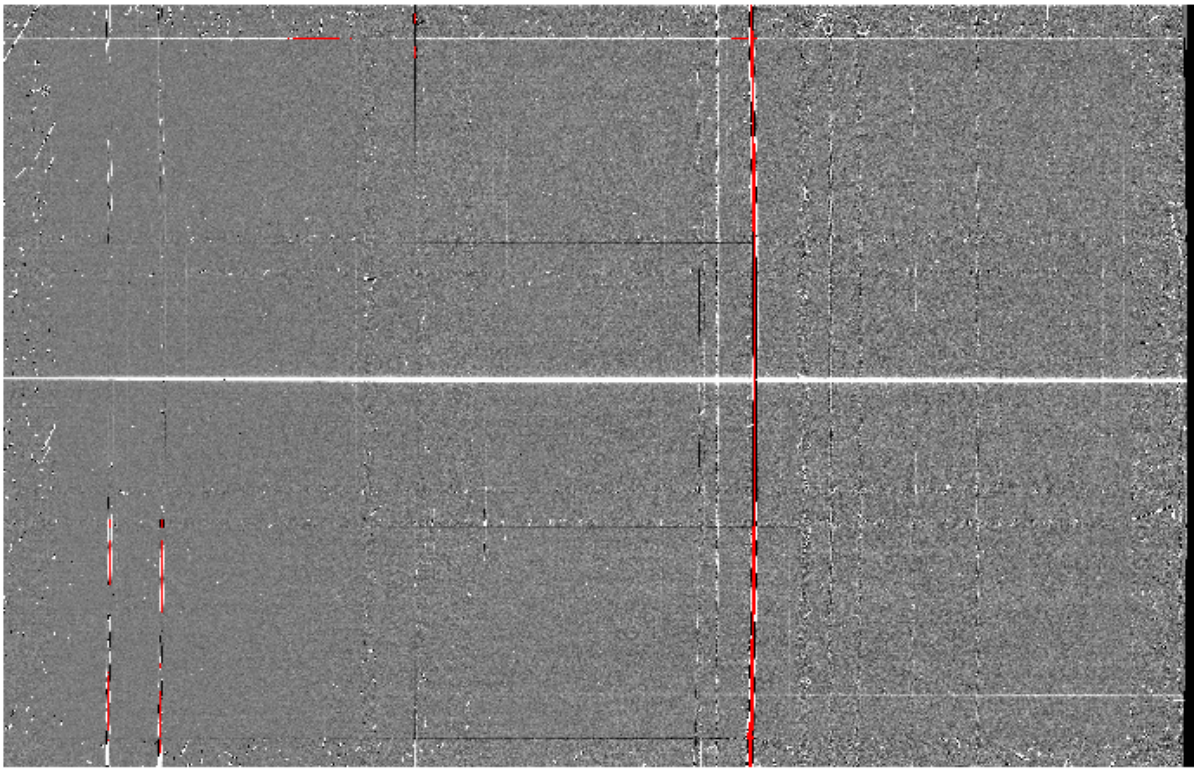
With the master bias, the master flat, the processed arcs (one for each of the grating position, aka central wavelength), and the processed standard in the local calibration manager, to reduce the science observations and extract the 1-D spectrum, one only needs to do as follows.

```
reduce_science = Reduce()
reduce_science.files.extend(scitarget)
reduce_science.mode = 'ql'
reduce_science.runr()
```

This produces a 2-D spectrum (`S20171022S0087_2D.fits`) which has been bias corrected, flat fielded, QE-corrected, wavelength-calibrated, corrected for distortion, sky-subtracted, and stacked. It also produces the 1-D spectrum (`S20171022S0087_1D.fits`) extracted from that 2-D spectrum. The 1-D spectrum is flux calibrated with the sensitivity function from the spectrophotometric standard. The 1-D spectra are stored as 1-D FITS images in extensions of the output Multi-Extension FITS file.

This is what the 2-D spectrum looks like.

```
display = Reduce()
display.files = ['S20171022S0087_2D.fits']
display.recipename = 'display'
display.runr()
```

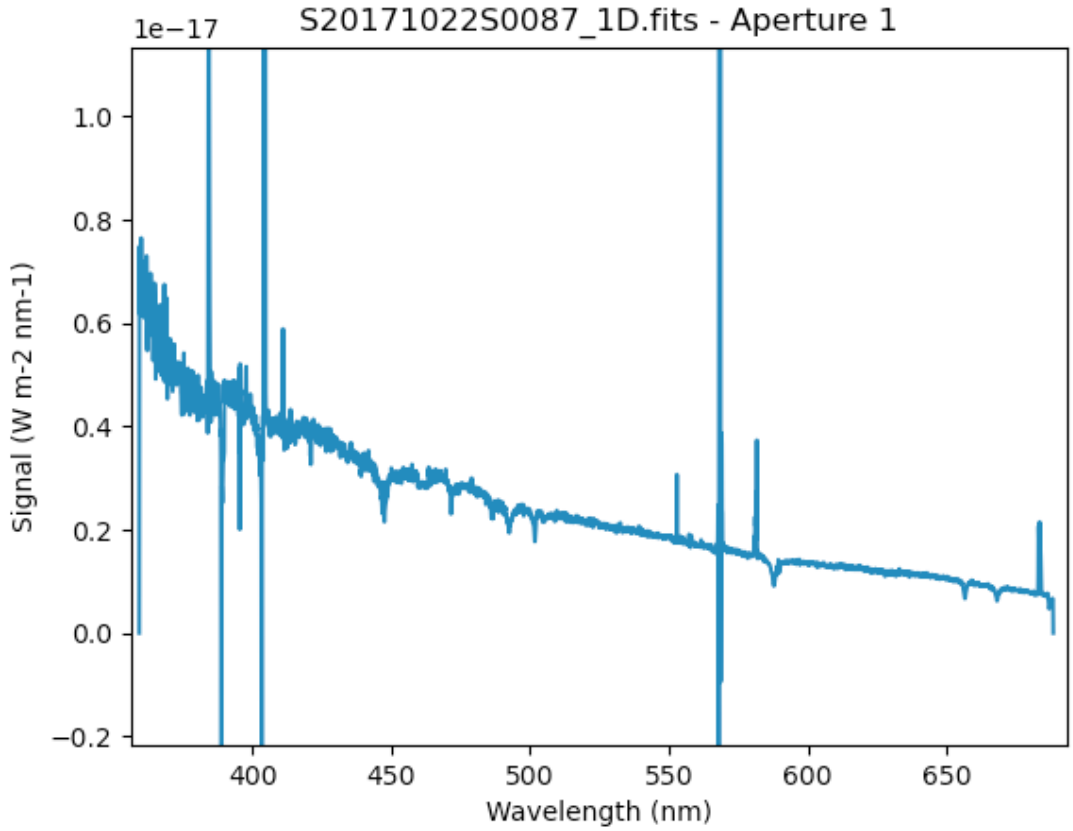


The apertures found are list in the log for the `findApertures` just before the call to `traceApertures`. Information about the apertures are also available in the header of each extracted spectrum: `XTRACTED`, `XTRACTLO`, `XTRACTHI`, for aperture center, lower limit, and upper limit, respectively.

This is what the 1-D flux-calibrated spectrum of our sole target looks like.

```
from gempy.adlibrary import plotting
import matplotlib.pyplot as plt

ad = astrodata.open(reduce_science.output_filenames[0])
plt.ioff()
plotting.dgsplot_matplotlib(ad, 1)
plt.ion()
```



To learn how to plot a 1-D spectrum with matplotlib using the WCS from a Python script, see Tips and Tricks *Plot a 1-D spectrum*.

If you need an ascii representation of the spectum, you can use the primitive `write1DSpectra` to extract the values from the FITS file.

```
writeascii = Reduce()
writeascii.files = ['S20171022S0087_1D.fits']
writeascii.recipename = 'write1DSpectra'
writeascii.runr()
```

The primitive outputs in the various formats offered by `astropy.Table`. To see the list, use **from the command line**.

```
showpars S20171022S0087_1D.fits write1DSpectra
```

To use a different format, set the `format` parameters.

```
writeascii = Reduce()
writeascii.files = ['S20171022S0087_1D.fits']
writeascii.recipename = 'write1DSpectra'
writeascii.uparms = [('format', 'ascii.ecsv'), ('extension', 'ecsv')]
writeascii.runr()
```

CHAPTER

FIVE

TIPS AND TRICKS

5.1 Plot a 1-D spectrum

Here is how to plot an extracted spectrum produced by DRAGONS with Python and matplotlib.

```
import matplotlib.pyplot as plt
import numpy as np

import astrodata
import gemini_instruments

ad = astrodata.open('S20171022S0087_1D.fits')
ad.info()

data = ad[0].data
wavelength = ad[0].wcs(np.arange(data.size)).astype(np.float32)
units = ad[0].wcs.output_frame.unit[0]

# add aperture number and location in the title.
# check that plt.xlabel call.  Not sure it's right, it works though.
plt.xlabel(f'Wavelength ({units})')
plt.ylabel(f'Signal ({ad[0].hdr["BUNIT"]})')
plt.plot(wavelength, data)
plt.show()
```

5.2 Inspect the sensitivity function

Plotting the sensitivity function is not obvious. Using Python, here's a way to do it.

```
from scipy.interpolate import BSpline
import numpy as np
import matplotlib.pyplot as plt

import astrodata
import gemini_instruments

ad = astrodata.open('S20170826S0160_ql_standard.fits')

sensfunc = ad[0].SENSFUNC
```

(continues on next page)

(continued from previous page)

```

order = sensfunc.meta['header'].get('ORDER', 3)
func = BSpline(sensfunc['knots'].data, sensfunc['coefficients'].data, order)
std_wave_unit = sensfunc['knots'].unit
std_flux_unit = sensfunc['coefficients'].unit

w1 = ad[0].wcs(0)
w2 = ad[0].wcs(ad[0].data.size)

x = np.arange(w1, w2)
plt.xlabel(f'Wavelength ({std_wave_unit})')
plt.ylabel(f'{std_flux_unit}')
plt.plot(x, func(x))
plt.show()
```

In the science-approved version of the GMOS longslit support in DRAGONS, there will be an interactive tool to inspect and adjust the sensitivity function.

CHAPTER

SIX

ISSUES AND LIMITATIONS

6.1 Double messaging issue

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

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

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search