European Organisation for Astronomical Research in the Southern Hemisphere

Programme: VLT

Project/WP: User Support Group

Reflex GIRAFFE Tutorial

Document Number: ESO-287261

Document Version: 1.5

Document Type: Manual (MAN)

Released on: 2020-07-30

Document Classification: ESO internal

Prepared by: J. Pritchard

Validated by:

Approved by:

Name

This page was intentionally left blank

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1.0	01/08/2016	All	First official release
1.1	06/02/2017	4, 6, 9 & 10	Revised for first implementation of GUIs, merging of Cookbook & Tutorial
1.2	20/04/2017	4, 6, 9 & 10	Revised for updated workflows
1.3	03/05/2019	4, 6, 9.8 & 10	Revised for improved science and standard star GUIs, sky subtraction
1.4	30/04/2020	6.2.4, 6.2.5	Section added.
1.5	30/07/2020	All	Reorganized and updated to new tutorial standards.

This page was intentionally left blank

Contents

1 Introduction to `Esoreflex` **9**

2 Software Installation **11**

2.1 Installing `Esoreflex` workflows via `macports` . . . 11

2.2 Installing `Esoreflex` workflows via `rpm/yum/dnf` . . . 11

2.3 Installing `Esoreflex` workflows via `install_esoreflex` . . . 11

2.4 Demo Data . . . 12

3 Quick Start: Reducing The Demo Data **13**

4 About the main `esoreflex` canvas **25**

4.1 Saving And Loading Workflows . . . 25

4.2 Buttons . . . 25

4.3 Workflow States . . . 25

5 The GIRAFFE Workflow **26**

5.1 Workflow Canvas Parameters . . . 26

5.2 Workflow Actors . . . 27

5.2.1 Simple Actors . . . 27

5.2.2 Composite Actors . . . 27

5.2.3 Recipe Execution within Composite Actors . . . 28

5.2.4 Adjusting any recipe parameter . . . 31

5.2.5 Lazy Mode . . . 32

6 Reducing your own data **33**

6.1 The `esoreflex` command . . . 33

6.2 Launching the workflow . . . 33

6.3 Workflow Steps . . . 35

6.3.1 Data Organisation And Selection . . . 35

6.3.2 DataSetChooser . . . 36

6.3.3 The ProductExplorer . . . 37

6.3.4 Creation of Master Calibration Files . . . 40
6.3.5 Response Computation . . . 40
6.3.6 Science Reduction . . . 40
6.3.7 Output Organisation . . . 41

7 Frequently Asked Questions **44**

8 Troubleshooting **47**

9 A brief overview of data reduction of multi-fiber spectroscopy data **49**
9.1 Multi-fiber spectroscopy . . . 49
9.2 Correcting detector cosmetic effects . . . 51
9.3 Fiber localization and tracing . . . 52
9.4 Extraction, flat-field spectra and fiber transmission . . . 53
9.5 Scattered light correction . . . 56
9.6 Wavelength calibration . . . 57
9.7 Extraction of the science . . . 57
9.8 Sky subtraction . . . 58

10 An extended Demo **61**
10.1 The Science and Standard GUIs in detail . . . 61
10.2 Cosmic Ray Cleaning . . . 61
10.2.1 PyCosmic . . . 63
10.2.2 Installing PyCosmic . . . 63
10.2.3 Astro-SCRAPPY . . . 63
10.2.4 Installing Astro-SCRAPPY . . . 64
10.2.5 Enabling Cosmic Ray Cleaning . . . 64
10.2.6 Cosmic Ray Cleaning results . . . 64
10.3 Sky Subtraction . . . 68
10.3.1 Known Limitations . . . 68
10.3.2 Method: basic description . . . 68
10.3.3 Method: error propagation . . . 70

10.3.4 Example 70
10.4 Creating a new *Slit Geometry Table* 72

11 IFU and ARGUS image reconstruction **73**
11.1 IFU and ARGUS image visualisation 73
11.2 Plate dependent IFU response 81

1 Introduction to Esoreflex

This document is a tutorial designed to enable the user to to reduce his/her data with the ESO pipeline run under an user-friendly environmet, called EsoReflex, concentrating on high-level issues such as data reduction quality and signal-to-noise (S/N) optimisation.

EsoReflex is the ESO Recipe Flexible Execution Workbench, an environment to run ESO VLT pipelines which employs a workflow engine to provide a real-time visual representation of a data reduction cascade, called a workflow, which can be easily understood by most astronomers. The basic philosophy and concepts of Reflex have been discussed by Freudling et al. (2013A&A...559A..96F). Please reference this article if you use Reflex in a scientific publication.

Reflex and the data reduction workflows have been developed by ESO and instrument consortia and they are fully supported. If you have any issue, please have a look to https://support.eso.org to see if this has been reported before or open a ticket for further support.

A workflow accepts science and calibration data, as downloaded from the archive using the CalSelector tool[1] (with associated raw calibrations) and organises them into DataSets, where each DataSet contains one science object observation (possibly consisting of several science files) and all associated raw and static calibrations required for a successful data reduction. The data organisation process is fully automatic, which is a major time-saving feature provided by the software. The DataSets selected by the user for reduction are fed to the workflow which executes the relevant pipeline recipes (or stages) in the correct order. Full control of the various recipe parameters is available within the workflow, and the workflow deals automatically with optional recipe inputs via built-in conditional branches. Additionally, the workflow stores the reduced final data products in a logically organised directory structure employing user-configurable file names.

This manual aims to supercede the *GIRAFFE data reduction cookbook* [2], which was last updated in 2009. The cookbook provided a general introduction to GIRAFFE data reduction concepts and guidance to use EsoRex with Gasgano and the command line to reduce GIRAFFE data. If you are determined to use Gasgano, then that manual may still be of use to you, but please bear in mind that some of the details described there (particularly details of the pipeline) are somewhat outdated now.

This manual focuses on data reduction using esoreflex and is structured in two parts. The first part is fairly generic and dedicated to esoreflex in general and the GIRAFFE esoreflex workflows in particular. The second part is an evolved form of the old cookbook providing a general introduction as well as practical advise related to data reduction beyond the basic processing provided by esoreflex.

The GIRAFFE esoreflex workflows described in the first part of this this document support the reduction of all standard mode GIRAFFE observations, i.e. Medusa, IFU and ARGUS data. The user is referred to the GIRAFFE user manual [3][2] for more information on the instrument itself, and the GIRAFFE pipeline user manual [4][3] for details of the GIRAFFE pipeline recipes.

There is (now) a single workflow, giraf.xml. It supports processing each science file individiually

[1] *https://www.eso.org/sci/archive/calselectorInfo.html*
[2] available at http://www.eso.org/sci/facilities/paranal/instruments/flames/doc.html
[3] available at http://eso.org/sci/software/pipelines/

and grouped. The workflow can group together either all the science from a single template execution, or all the science files of a given target with the same setup and using the same plate during a given night. Selection between the different grouping strategies is controlled via a variable in the workflow main canvas.

The quick start section (see Section 3) describes the minimum effort required to get started, and it makes up only 5 pages of text in this tutorial.

In the second part of this document (begining at section 9) we provide a general introduction to GIRAFFE data and its reduction and then *attempt* to describe at least *some* of the additional steps required toward a full and complete scientific reduction of GIRAFFE data beyond that provided by `esoreflex` including treatment of sky subtraction, co-addition of multiple exposures, and further treatment of some of the issues identified in the tutorial.

2 Software Installation

`Esoreflex` and the workflows can be installed in different ways: via package repositories, via the `install_esoreflex` script or manually installing the software tar files.

The recommended way is to use the package repositories if your operating system is supported. The pipelines and Reflex can be installed from the ESO `macports` repositories that support macOS platforms, the and the `rpm/yum` repositories that support Fedora and CentOS platforms. For any other operating system it is recommended to use the `install_esoreflex` script.

The installation from package repository requires administrative privileges (typically granted via sudo), as it installs files in system-wide directories under the control of the package manager. If you want a local installation, or you do not have sudo privileges, or if you want to manage different installations on different directories, then use the `install_esoreflex` script. Note that the script installation requires that your system fulfill several software prerequisites, which might also need sudo privileges.

Reflex 2.11.x needs java JDK 11 to be installed.

Please note that in case of major or minor (affecting the first two digit numbers) Reflex upgrades, the user should erase the `$HOME/KeplerData`, `$HOME/.kepler` directories if present, to prevent possible aborts (i.e. a hard crash) of the `esoreflex` process.

2.1 Installing `Esoreflex` workflows via `macports`

This method is supported for the macOS operating system. It is assumed that macports (*https://www.macports.org*) is installed. Please read the full documentation at *https://www.eso.org/sci/software/pipelines/installation/macports.html*, which also describes the versions of macOS that are currently supported.

2.2 Installing `Esoreflex` workflows via `rpm/yum/dnf`

This method is supported for Fedora and CentOS platforms and requires sudo rights. Please read the full documentation at *https://www.eso.org/sci/software/pipelines/installation/rpm.html*, which also describes the versions of Fedora and CentOS that are currently supported.

2.3 Installing `Esoreflex` workflows via `install_esoreflex`

This method is recommended for operating systems other than what indicated above, or if the user has no sudo rights. Software dependencies are not fulfilled by the installation script, therefore the user has to install all the prerequisites before running the installation script.

The software pre-requisites for `Reflex 2.11.0` may be found at:
https://www.eso.org/sci/software/pipelines/reflex_workflows

To install the `Reflex 2.11.0` software and demo data, please follow these instructions:

1. From any directory, download the installation script:

```
wget https://eso.org/sci/software/pipelines/install_esoreflex
```

2. Make the installation script executable:

```
chmod u+x install_esoreflex
```

3. Execute the installation script:

```
./install_esoreflex
```

   and the script will ask you to specify three directories: the download directory `<download_dir>`, the software installation directory `<install_dir>`, and the directory to be used to store the demo data `<data_dir>`. If you do not specify these directories, then the installation script will create them in the current directory with default names.

4. Follow all the script instructions; you will be asked whether to use your Internet connection (recommended: yes), the pipelines and demo-datasets to install (note that the installation will remove all previously installed pipelines that are found in the same installation directory).

5. To start `Reflex`, issue the command:

```
<install_dir>/bin/esoreflex
```

   It may also be desirable to set up an alias command for starting the `Reflex` software, using the shell command `alias`. Alternatively, the `PATH` variable can be updated to contain the `<install_dir>/bin` directory.

2.4 Demo Data

Together with the pipeline you will also receive a demo data set, that allows you to run the `Reflex GIRAFFE` workflow without any changes in parameters for one data set and then leads you through exploration of the products and parameter adjustments you may need to reduce your data. This way you have a data set to experiment with before you start to work on your own data.

Note that you will need a minimum of ∼0.5 GB, ∼0.6 GB and ∼1 GB of free disk space for the directories `<download_dir>`, `<install_dir>` and `<data_dir>`, respectively if you are installing from the kit.

The GIRAFFE demo data have been retrieved with the CalSelector tool[4].

[4] http://www.eso.org/sci/archive/calselectorInfo.html

3 Quick Start: Reducing The Demo Data

For the user who is keen on starting reductions without being distracted by detailed documentation, we describe the steps to be performed to reduce the science data provided in the GIRAFFE demo data set supplied with the `esoreflex 2.11.0` release. By following these steps, the user should have enough information to perform a reduction of his/her own data without any further reading:

1. First, type:

   ```
   esoreflex -l
   ```

   If the `esoreflex` executable is not in your path, then you have to provide the command with the executable full path `<install_dir>/bin/esoreflex -l` . For convenience, we will drop the reference to `<install_dir>`. A list with the available `esoreflex` workflows will appear, showing the workflow names and their full path.

2. Open the GIRAFFE by typing:

   ```
   esoreflex giraf&
   ```

   Alternatively, you can type only the command `esoreflex` the empty canvas will appear (Figure 3.12) and you can select the workflow to open by clicking on `File -> Open File`. Note that the loaded workflow will appear in a new window. The GIRAFFE workflow is shown in Figure 3.1.

3. To aid in the visual tracking of the reduction cascade, it is advisable to use component (or actor) highlighting. Click on `Tools -> Animate at Runtime`, enter the number of milliseconds representing the animation interval (100 ms is recommended), and click `OK`.

4. Change directories set-up. Under "Setup Directories" in the workflow canvas there are seven parameters that specify important directories (green dots).

   By default, the `ROOT_DATA_DIR`, which specifies the working directory within which the other directories are organised. is set to your `$HOME/reflex_data` directory. All the temporary and final products of the reduction will be organized under sub-directories of `ROOT_DATA_DIR`, therefore make sure this parameter points to a location where there is enough disk space. To change `ROOT_DATA_DIR`, double click on it and a pop-up window will appear allowing you to modify the directory string, which you may either edit directly, or use the `Browse` button to select the directory from a file browser. When you have finished, click `OK` to save your changes.

   Changing the value of `RAW_DATA_DIR` is the only necessary modification if you want to process data other than the demo data

5. Click the button to start the workflow

6. The workflow will highlight the `Data Organiser` actor which recursively scans the raw data directory (specified by the parameter `RAW_DATA_DIR` under "Setup Directories" in the workflow canvas) and constructs the datasets. Note that the raw and static calibration data must be present either in `RAW_DATA_DIR` or in `CALIB_DATA_DIR`, otherwise datasets may be incomplete and cannot be processed. However, if the same reference file was downloaded twice to different places this creates a problem as `esoreflex` cannot decide which one to use.

7. The `Data Set Chooser` actor will be highlighted next and will display a "Select Datasets" window (see Figure 3.2) that lists the datasets along with the values of a selection of useful header keywords[5]. The first column consists of a set of tick boxes which allow the user to select the datasets to be processed. By default all complete datasets which have not yet been reduced will be selected. A full description of the options offered by the `Data Set Chooser` will be presented in Section 6.3.2.

8. Click the `Continue` button and watch the progress of the workflow by following the red highlighting of the actors. A window will show which dataset is currently being processed.

[5]The keywords listed can be changed by double clicking on the `DataOrganiser` Actor and editing the list of keywords in the second line of the pop-up window. Alternatively, instead of double-clicking, you can press the right mouse button on the `DataOrganiser` Actor and select `Configure Actor` to visualize the pop-up window.

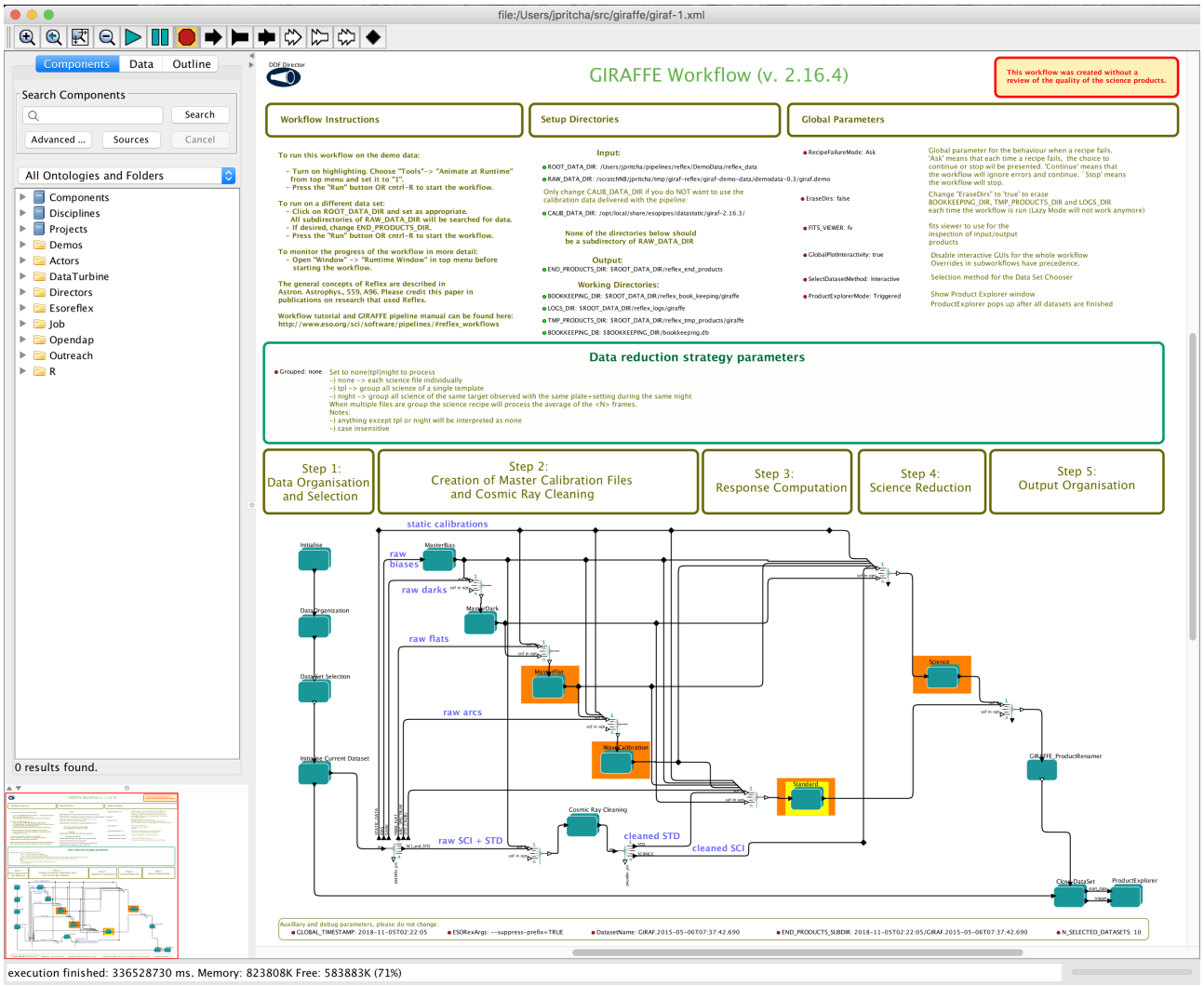


Figure 3.1: *General layout of the 'GIRAFFE Workflow :: One-by-One' workflow..*

9. The workflow will commence the processing of the first dataset.

   Interactive GUIs have been implemented for the MasterFlat, WaveCalibration, Standard and Science actors[6], these are highlighted by the orange rectangles around the respective actors in the workflow canvas. Whether each Interactive GUI is enabled or not can be controlled individually through the 'EnableInteractivity' variable of each Actor. The 'EnableInteractivity' variable can be edited in the 'Edit Parameters' window of each actor[7] see figure 3.3.

   By default all GUIs are enabled in the workflows contained in the distribution. As can be seen in figure 3.3 the EnableInteractivity variable is set to the variable `$GlobalPlotInteractvity`

[6]Each of the green icons in the workflow canvas is referred to as an 'actor'.

[7]The 'Edit Parameters' window of an actor can be accessed either by right-button clicking on the actor and then selecting 'Configire Actor' from the pop-up menu, by selecting the actor and the typing the short-cut as indicated in the afore-mentioned pop-up menu, or simply by double clicking on the actor.

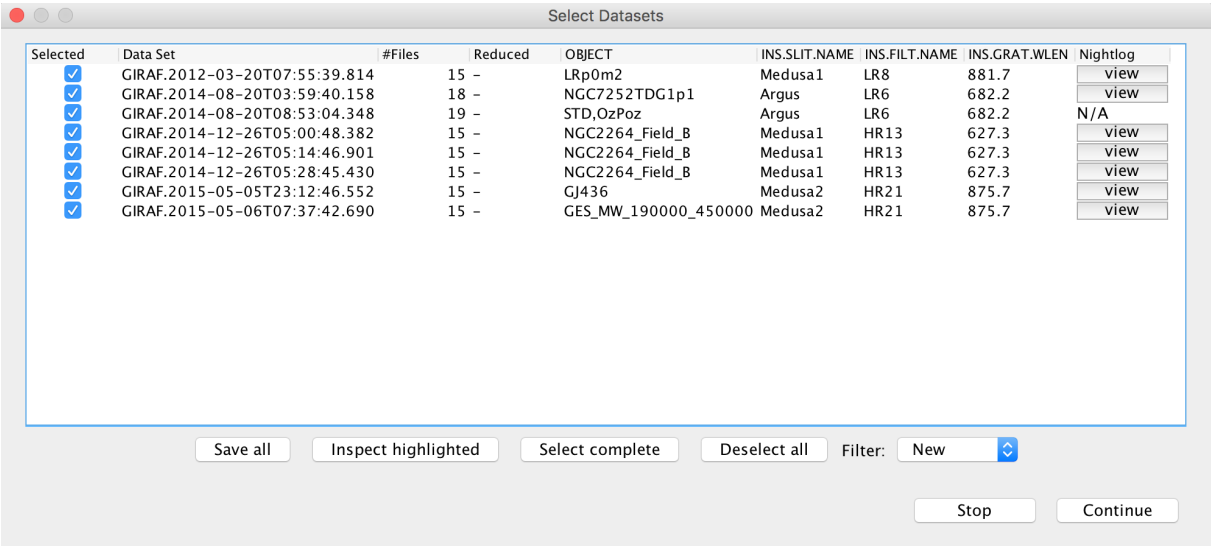


Figure 3.2: *The 'Select Datasets' pop-up window. Note it may be necessary to increase the width of the window in order to see all the information.*

which is defined and set in the 'Global Parameters' section of the workflow main canvas. All the Interactive GUIs can be disabled simply by setting GlobalPlotInteractvity to false, or the GUIs can be enabled and disabled individually via the EnableInteractivity variable of each actor.

For the purposes of this tutorial we assume that you leave interactivity enabled for all the actors.

The workflow will then proceed with execution of the BIAS actor, followed quickly by the 'Cosmic ray Cleaning' actor[8] and then followed by the MasterFlat actor, the first actor with an interactive GUI. After executing the pipeline recipe, the GUI window will open. You should see a window like that in figure 3.4. It has the following features:

- Standard MatPlotLib graphical interactivity tools allowing zooming, panning etc in the top left of the window[9]
- A file selector (at the bottom of the window) allowing the selection between displaying the PSF fitting or the order localization.
- A display of the MasterFlat image with the traced orders over-plotted in green and blue colours on the PSF fitting and red and yellow colours on the localization tracing, alternating between the sub-slit groupings of the fibres. A black line is also drawn across the image representing the row that was used for the initial fibre localization, and just above this the indiviual fibres are labeled with their FPS number. At the top of the image the sub-slits are labeled 1 to 13.
- Below this is a plot across the row used for the initial fibre localization, and again the measured position of each fibre in this row is indicated by a blue or green/red or yellow

[8]This actor principally performs cosmic ray cleaning if/when the appropriate additional/non-standard python modules (AstroScrappy and/or PyCosmic) are installed and available to esoreflex, see section 10.2.

[9]The exact appearance of the window may not quite match the one shown in this tutorial if you have not yet upgraded to MatPlotLib 2, but in general everything should work analagously for pre version 2 MatPlotLib systems.

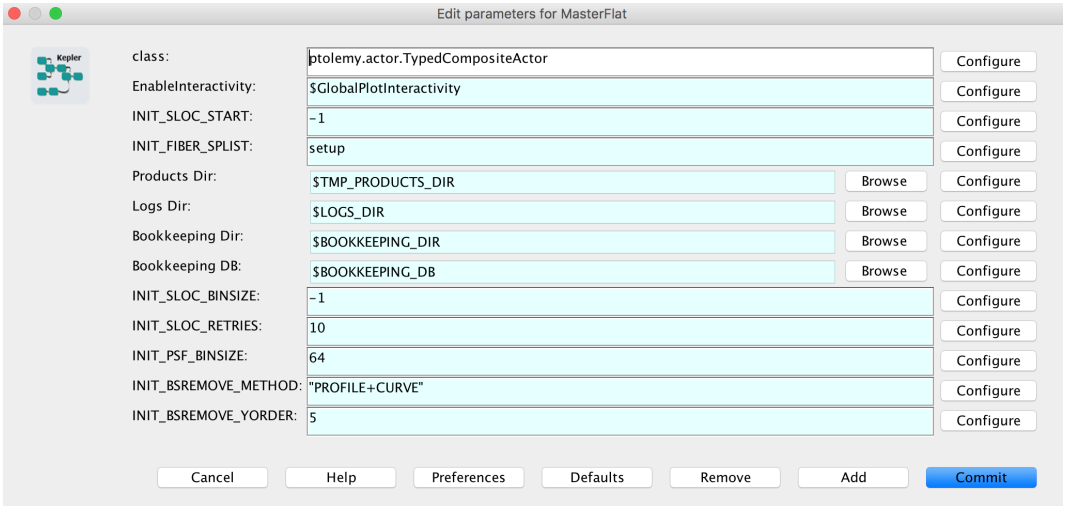


Figure 3.3: *The 'Edit Parameters' window for the MasterFlat actor.*

vertical line, again alternating between the sub-slit groupings of the fibres. Also again the individual fibres FPS numbers and sub-slit numbers are labeled at the bottom and top of the plot window.

- below this various information about the raw data is presented.
- On the right of the window are fields, grouped into tabs by function where recipe parameters can be adjusted 'on-the-fly', buttons to either continue the workflow, re-run the recipe (usually after having adjusted some pipeline parameters), display the esorex log file in a text window, or display some help text in a window. There are also two check boxes below the buttons that allow one to control whether the interactive window is displayed in further executions of this actor and to signal that the current values of the parameters in the fields above should be used as the initial parameters for all future executions of this actor in this workflow. Finally there is a 'Files info' widget, click on the small triangle just in front of 'Files info' to display a list of the input files used by the recipe and the output files generated by the recipe.

10. **Interact with the displays**: click on the MatPlotLib Zoom button, [10] to enter Zoom mode. Now click and hold the mouse button while over the image or cross-cut display subplots and then drag the mouse to define a rectangular region (see figure 3.5) which will be redisplayed zoomed into the respective display window (see figure 3.6).

    Click the MatPlotLib Pan button, to enter panning mode. Now you can click-hold-drag the mouse button to recenter the image at the zoomed view. Click the backward, and forward, buttons to navigate through the various views you have set, or click the Home button, to return to the default full chip view.

[10] On pre MatPlotLib ver 2.0 systems the zoom button looks like , the Pan button looks like , the Back button looks like , the Forward button looks like , and the Home button looks like ,

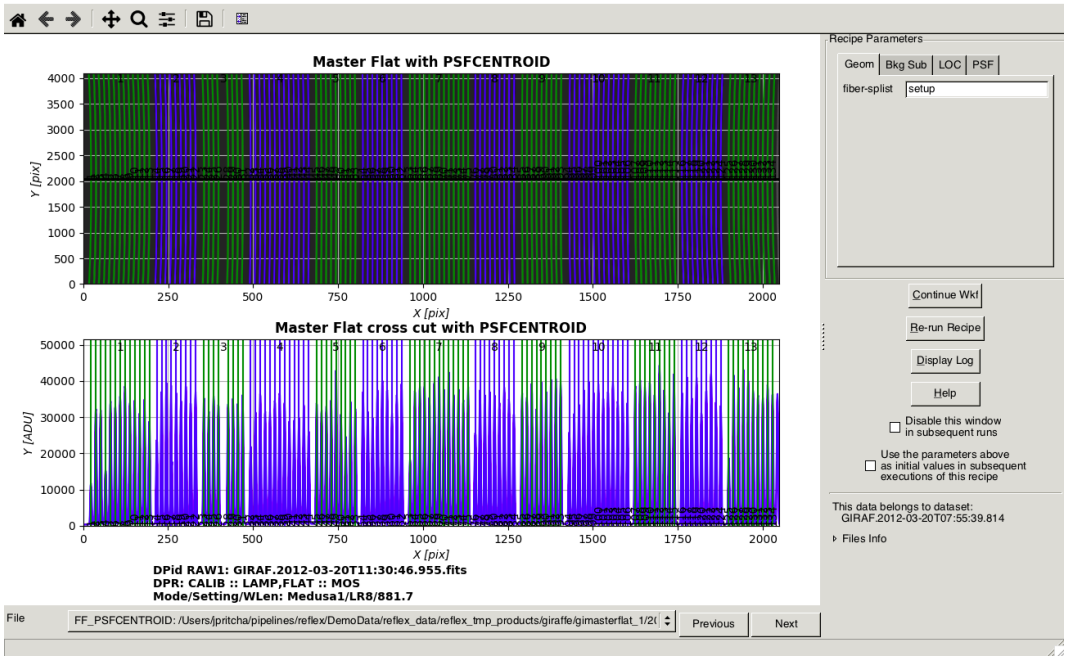


Figure 3.4: *The MasterFlat Interactive GUI for the first dataset. Like all standard Reflex GUIs it is based on a standard MatPlotLib widget set, hence you see the standard set of controls allowing zooming, panning, adjusting the borders and spacing and saving the image to disk, albeit in the unusual position of the top of the screen. The other features of the GUI are described in the text.*

Feel free to continue exploring the displayed images. When you've had enough, continue to the next step.

11. **Adjust recipe parameters**: Let's begin with something simple, but clear. In the `fiber-splist` field, repalce the text 'setup' with '1-100'. Then click the the `Re-run Recipe` button. The GUI window will close and the workflow will re-run the MasterFlat recipe. Eventually the GUI window will re-open, but only the first 100 fibres will now be traced, see figure 3.7.

    Other parameters can be adjusted as easily and their effects seen as easily as clicking the `Re-run Recipe` button and then waiting for the GUI to redisplay the results.

    The GUI presents only a limited subset of parameters for adjustment, those found to be of use in solving some of the problems in the data thus far encountered. If it is necessary to adjust other parameter(s), you must stop the workflow, open the actor, and then edit the parameters of the recipe actor contained inside the actor. This is beyond the scope of this quickstart, but described in section 5.2.4.

    The majority of data, especially Medusa mode, reduce well with the default parameter set, so normally there is not much need to adjust parameters. So for now, set the `fiber-splist` parameter back to 'setup', click the `Re-run Recipe` button and when the MasterFlat GUI reappears click the `Continue Wkf` button.

12. The workflow will now proceed to the next pipeline stage, Wavelength Calibration.

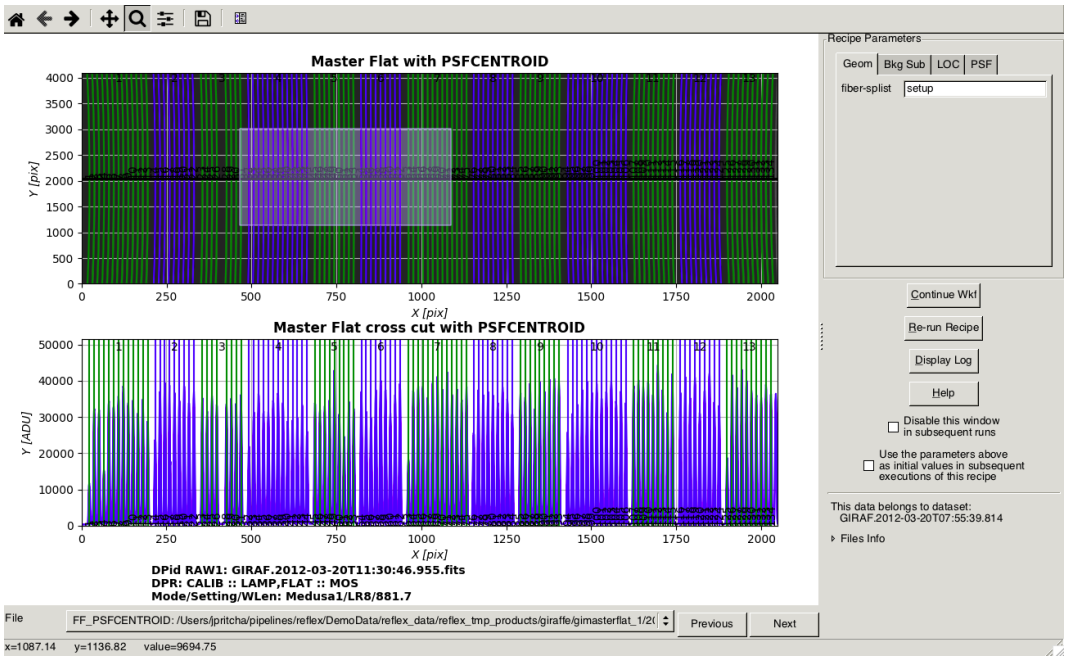


Figure 3.5: *Zooming in on the displayed MasterFlat.*

After running the recipe, the Wavecalibration interactive GUI will open, see figure 3.8.

The interactivity in this window (as in all the interactive GUI windows) is analogous to that described for the MasterFlat GUI.

The WaveCalibration step is generally robust and seldom shows problems. However occassionally there can be problems. If the displayed image looks more like figure 10.6 the the solution is to create a new *slit geometry table*, see section 10.4.

When ready click, the `Continue Wkf` button.

13. The workflow will now proceed to the Science actor and after running the recipe open the Science interactive GUI, see figure 3.9.

    The Science GUI features:

    - On the right hand side of the panel the usual GUI controls, as described for the MasterFlat GUI, i.e. parameter adjustment, continue and re-run buttons etc.
    - In the upper part of the panel, an image display of the 2-D rebinned spectrum image, each column representing one fibre and the rows the spectral dispersion.
    - As you move the mouse over the 2-D image, the x and y position and the pixel value, and the object name, the mean S/N of that spectrum and the FPS/SSN and column numbers of the spectrum under the mouse are dispalyed in the status bar at the bottom of the window.
    - The file selector at the bottom of the window allows you to select which version of the 2-D rebinned spectrum to display. You can choose between the different reductions with and without Cosmic Ray correction if the Cosmic Ray correction packages are installed.

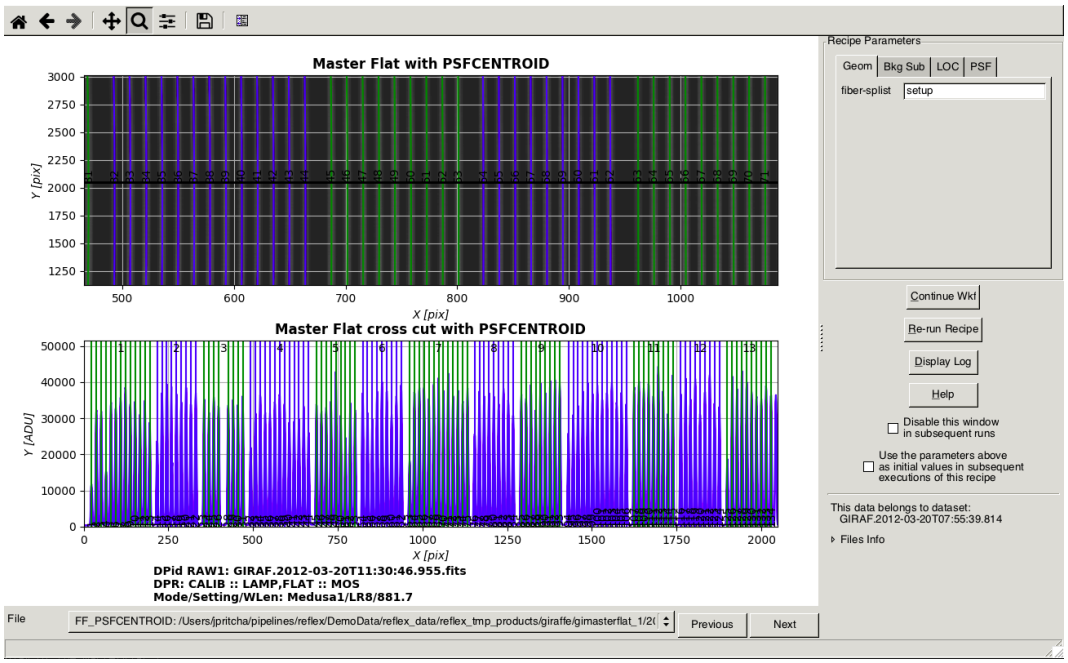


Figure 3.6: *The zoomed display of the MasterFlat.*

- In the middle of the panel, a spectrum display for plots of 1-D spectra for the fibre. The object name, S/N ratio, FPS and SSN numbers of the fibre and the column number in the 2-D image above are displayed in the title of the plot.
- Below the spectrum display, a series of list-selectors, buttons, and radio-buttons which allow you to select which fibre is being displayed, which versions[11] of the processing are displayed, whether to display the signal, signal-sky, sky and/or errors in the spectrum display. There are also buttons to control the various options for the sky-subtraction. One is able to simultaneously display the different versions of the reduction/sky-subtraction so that one can directly compare them.
- Changes to the sky-subtraction parameters cause the sky to be re-computed 'on-the-fly' and then the spectrum display is updated in realtime. The functioning of the sky subtraction is discussed in greater detail in sections 9.8 and section 10.3.
- When changing from one fibre to another, any zooming or panning of the spectrum display window is preserved allowing one to compare directly the details of different fibres.

Again you can use the standard MatPlotLib functionalities to exploere the 2-D image and 1-D spectrum, e.g. see figure 3.10 for a zoomed view around the absoption line centered at approximately 866nm.

14. **Plot selected fibres**: You can change the plotted fibre in the 1-D spectrum panel by clicking on the 2-D image with the middle mouse button[12]. By default, the science GUI window displays the object with the highest measured S/N when it opens.

[11] No-cosmic-ray-correction, the cosmic ray corrected and medianed
[12] On a Mac you may need to install a utiltiy such as MiddleClick, http://http://rouge41.com/labs/

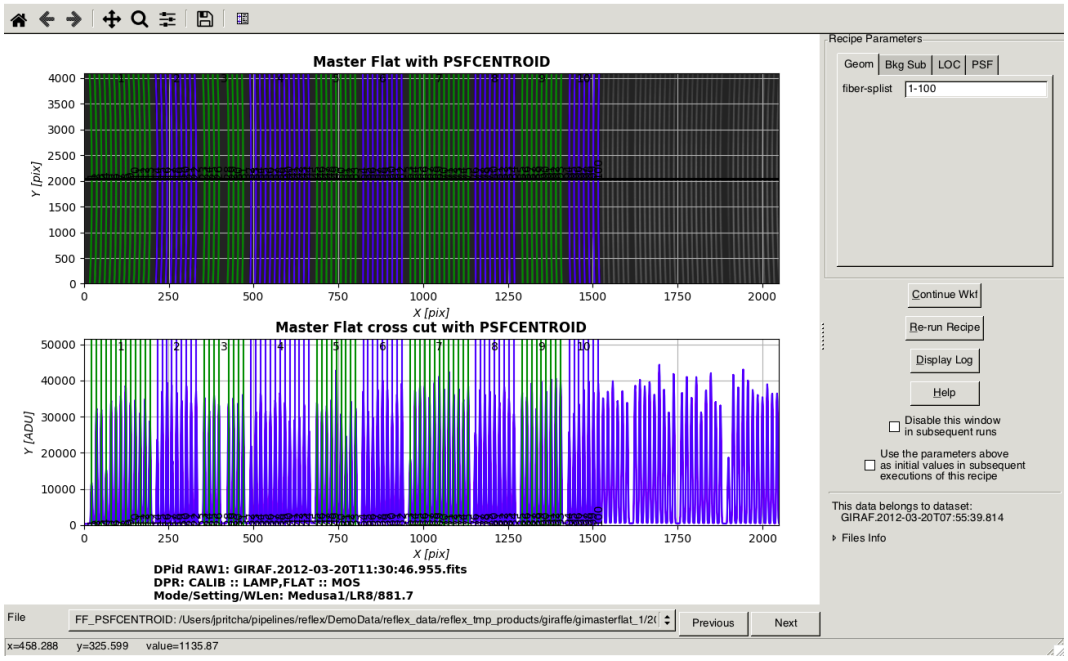


Figure 3.7: *The MasterFlat Interactive GUI for the first dataset, after re-running the recipe with* `fiber-splist` *set to '1-100'.*

When ready click, the `Continue Wkf` button.

15. The workflow will now continue with the remaining datasets.

16. The 2nd dataset is an ARGUS observation with low S/N. Currently there is no display for the reconstructed cube image. Please note that this dataset includes two MasterFlat datasets, one for the daytime FibreFlat, `DPR.TYPE = LAMP,FLAT` (associated to the Wave Calibration), and one for the Nasmyth Flat, `DPR.TYPE = LAMP,FLAT,NASMYTH` taken immediately after and associated to the ARGUS Science exposure.

17. The 3rd dataset is the ARGUS STD star observation associated with the previous ARGUS observation. This spectrum shows a high S/N spectrum of a star spread over multiple fibres. Currently there is no handling in the workflow for flux calibration. This dataset also includes two MasterFlat datasets, as above.

18. The 4th dataset is the IFU STD star observation associated with the following IFU observation. All 13 IFUs are enabled but there is a standard star on only one of them.

19. The 5th dataset is an IFU science observation with objects centred on five of the IFUs.

20. The 6th, 7th and 8th datasets are three exposures acquired with a single template, so if run through the GRP workflow, they will be processed as a single dataset with the Science recipe computing the final 1-D spectra from the mean of the three input spectra.

21. The 9th dataset is a calibrator observed for the Gaia-ESO public survey in Medusa mode. There is only ony fibre assigned to a star, the others are all sky fibres.

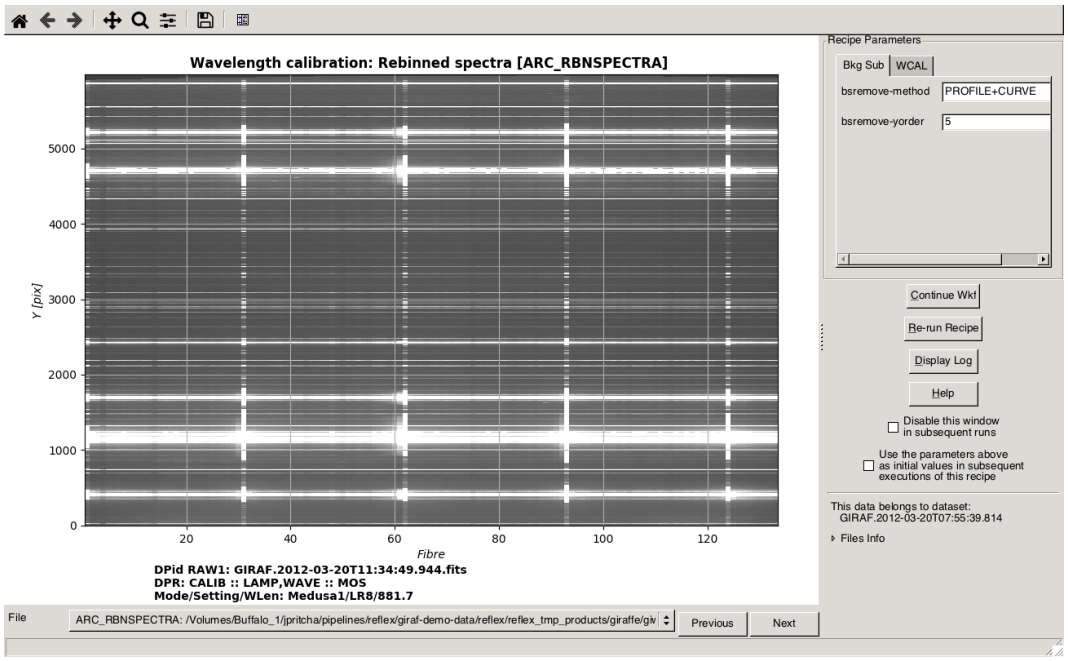


Figure 3.8: *The WaveCalibration Interactive GUI for the first dataset.*

22. The 10th dataset is a randomly selected science field from Gaia-ESO.

23. Once the reduction of all datasets has finished, a pop-up window called *Product Explorer* will appear, showing the datasets which have been reduced together with the list of final products. This actor allows the user to inspect the final data products, as well as to search and inspect the input data used to create any of the products of the workflow. Figure 3.11 shows the Product Explorer window. A full description of the *Product Explorer* will be presented in Section 6.3.3.

24. After the workflow has finished, all the products from all the datasets can be found in a directory under `END_PRODUCTS_DIR` named after the workflow start timestamp. Further subdirectories will be found with the name of each dataset.

Well done! You have successfully completed the quick start section and you should be able to use this knowledge to reduce your own data. However, there are many interesting features of `Reflex` and the GIRAFFE workflow that merit a look at the rest of this tutorial.

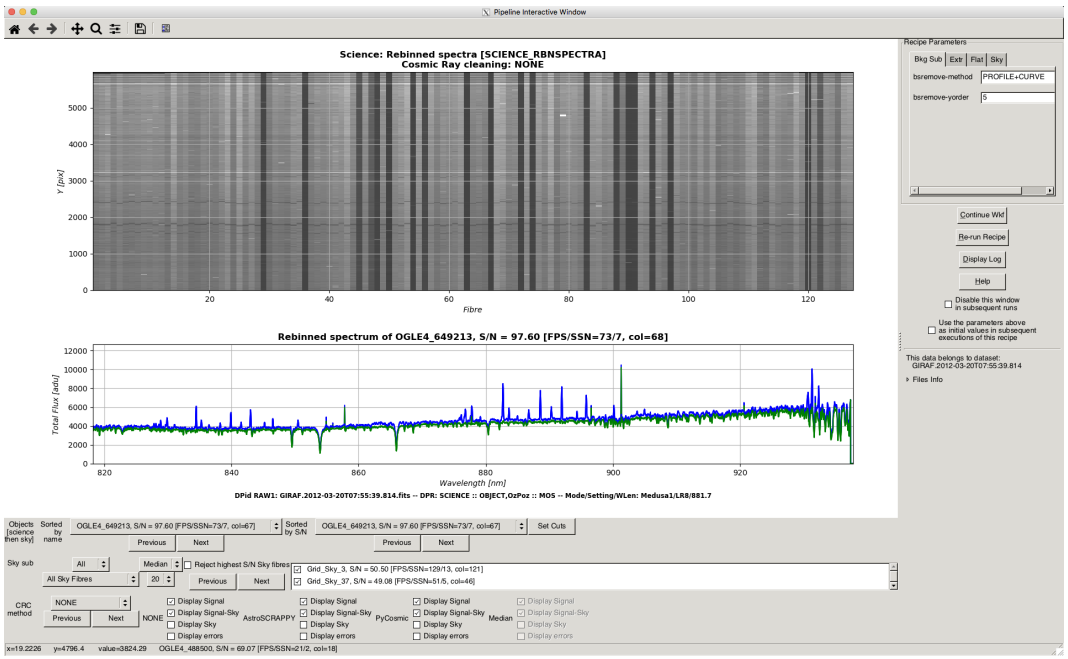


Figure 3.9: *The Science Interactive GUI for the first dataset. The blue lines are the 'Signal', the green lines are the 'Signal-Sky'.*

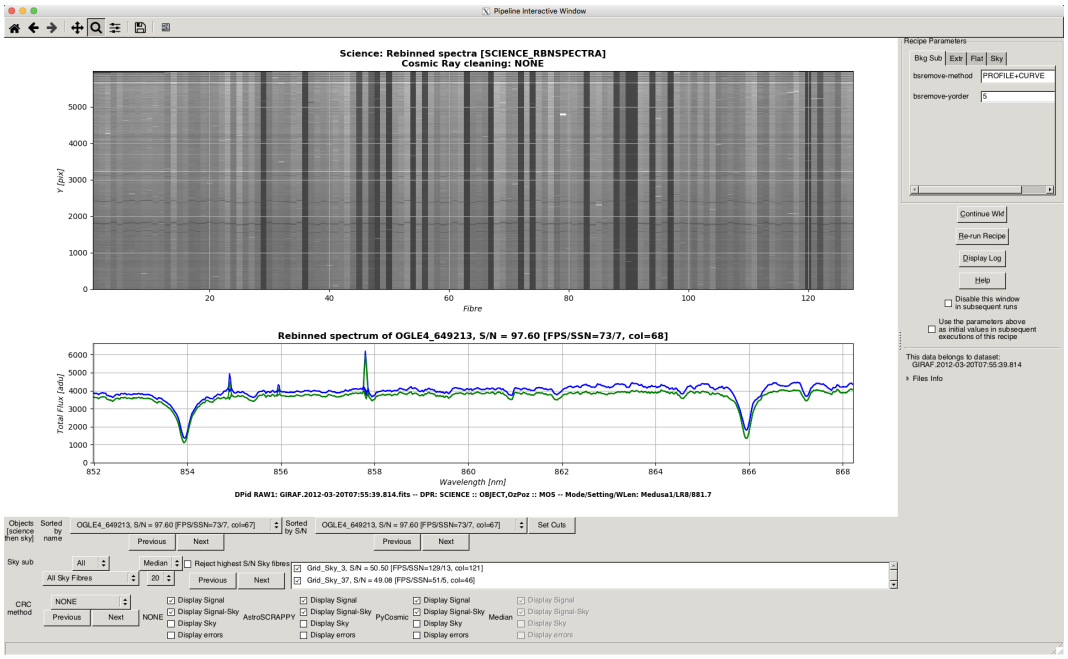


Figure 3.10: *The Science Interactive GUI for the first dataset with a selected zoom in the displayed spectrum. The blue lines are the 'Signal', the green lines are the 'Signal-Sky'. In a few places you can see the differences between the Cosmic Ray Cleaned spectra and the non-cleaned spectrum, and even between the two Cosmic Ray Cleaned spectra themselves.*

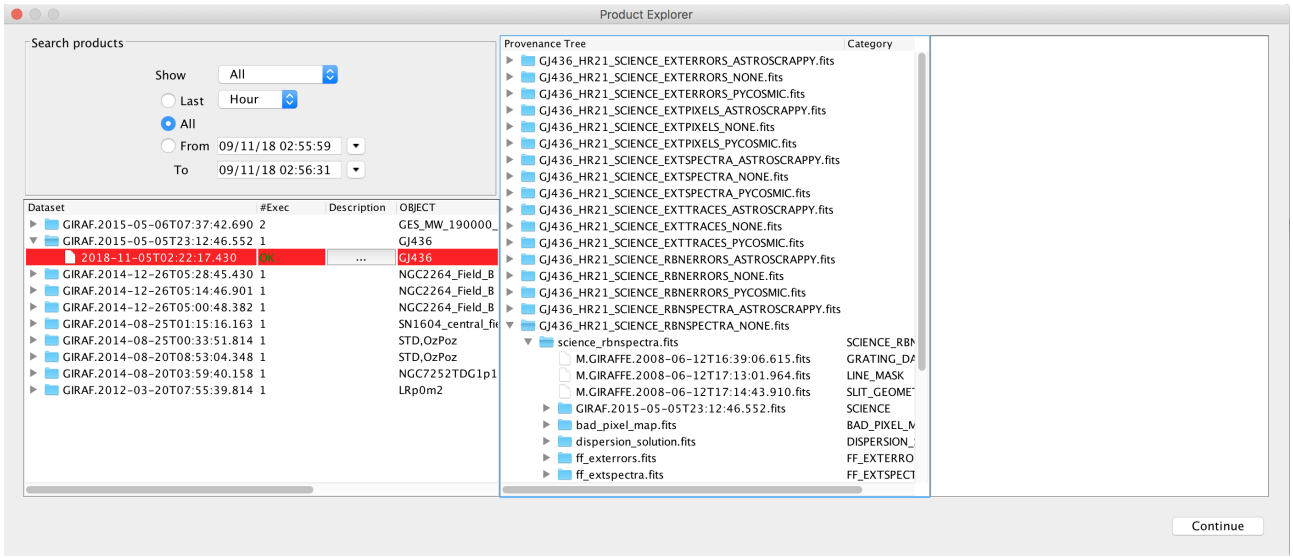


Figure 3.11: *The Provenance Explorer shows all datasets reduced in previous executions together with the full reduction chain for all the pipeline products.*

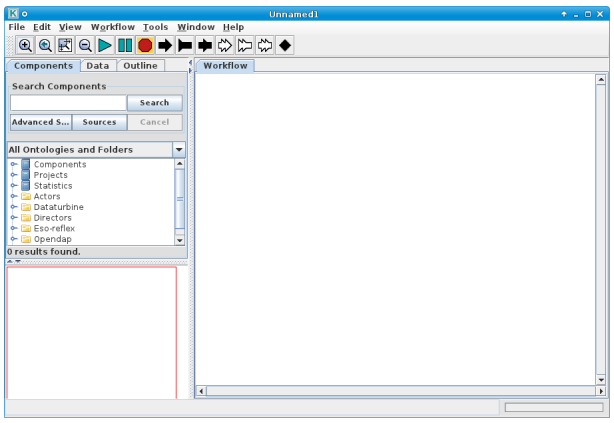


Figure 3.12: *The empty* `Reflex` *canvas.*

4 About the main `esoreflex` canvas

4.1 Saving And Loading Workflows

In the course of your data reductions, it is likely that you will customise the workflow for various data sets, even if this simply consists of editing the `ROOT_DATA_DIR` to a different value for each data set. Whenever you modify a workflow in any way, you have the option of saving the modified version to an `XML` file using `File -> Export As` (which will also open a new workflow canvas corresponding to the saved file). The saved workflow may be opened in subsequent `esoreflex` sessions using `File -> Open`. Saving the workflow in the default Kepler format (.kar) is only advised if you do not plan to use the workflow with another computer.

4.2 Buttons

At the top of the `esoreflex` canvas are a set of buttons which have the following functions:

- - Zoom in.
- - Reset the zoom to 100%.
- - Zoom the workflow to fit the current window size (Recommended).
- - Zoom out.
- - Run (or resume) the workflow.
- - Pause the workflow execution.
- - Stop the workflow execution.

The remainder of the buttons (not shown here) are not relevant to the workflow execution.

4.3 Workflow States

A workflow may only be in one of three states: executing, paused, or stopped. These states are indicated by the yellow highlighting of the , , and buttons, respectively. A workflow is executed by clicking the button. Subsequently the workflow and any running pipeline recipe may be stopped immediately by clicking the button, or the workflow may be paused by clicking the button which will allow the current actor/recipe to finish execution before the workflow is actually paused. After pausing, the workflow may be resumed by clicking the button again.

5 The GIRAFFE Workflow

The GIRAFFE workflow canvas is organised into a number of areas. From top-left to top-right you will find general workflow instructions, directory parameters, and global parameters. In the middle row you will find five boxes describing the workflow general processing steps in order from left to right, and below this the workflow actors themselves are organised following the workflow general steps.

5.1 Workflow Canvas Parameters

The workflow canvas displays a number of parameters that may be set by the user. Under "Setup Directories" the user is only required to set the `RAW_DATA_DIR` to the working directory for the dataset(s) to be reduced, which, by default, is set to the directory containing the demo data. The `RAW_DATA_DIR` is recursively scanned by the `Data Organiser` actor for input raw data. The directory `CALIB_DATA_DIR`, which is by default within the pipeline installation directory, is also scanned by the `Data Organiser` actor to find any static calibrations that may be missing in your dataset(s). If required, the user may edit the directories `BOOKKEEPING_DIR`, `LOGS_DIR`, `TMP_PRODUCTS_DIR`, and `END_PRODUCTS_DIR`, which correspond to the directories where book-keeping files, logs, temporary products and end products are stored, respectively (see the Reflex User Manual for further details; [8]).

There is a mode of the `Data Organiser` that skips the built-in data organisation and uses instead the data organisation provided by the CalSelector tool. To use this mode, click on `Use CalSelector associations` in the `Data Organiser` properties and make sure that the input data directory contains the XML file downloaded with the CalSelector archive request (note that this does not work for all instrument workflows).

Under the "Global Parameters" area of the workflow canvas, the user may set the `FITS_VIEWER` parameter to the command used for running his/her favourite application for inspecting FITS files. Currently this is set by default to `fv`, but other applications, such as `ds9`, `skycat` and `gaia` for example, may be useful for inspecting image data. Note that it is recommended to specify the full path to the visualization application (an alias will not work).

By default the `EraseDirs` parameter is set to `false`, which means that no directories are cleaned before executing the workflow, and the recipe actors will work in Lazy Mode (see Section 5.2.5), reusing the previous pipeline recipe outputs if input files and parameters are the same as for the previous execution, which saves considerable processing time. Sometimes it is desirable to set the `EraseDirs` parameter to `true`, which forces the workflow to recursively delete the contents of the directories specified by `BOOKKEEPING_DIR`, `LOGS_DIR`, and `TMP_PRODUCTS_DIR`. This is useful for keeping disk space usage to a minimum and will force the workflow to fully re-reduce the data each time the workflow is run.

The parameter `RecipeFailureMode` controls the behaviour in case that a recipe fails. If set to `Continue`, the workflow will trigger the next recipes as usual, but without the output of the failing recipe, which in most of the cases will lead to further failures of other recipes without the user actually being aware of it. This mode might be useful for unattended processing of large number of datasets. If set to `Ask`, a pop-up window will ask whether the workflow should stop or continue. This is the

default. Alternatively, the `Stop` mode will stop the workflow execution immediately.

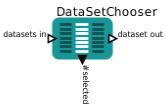
The parameter `ProductExplorerMode` controls whether the `ProductExplorer` actor will show its window or not. The possible values are `Enabled`, `Triggered`, and `Disabled`. `Enabled` opens the ProductExplorer GUI at the end of the reduction of each individual dataset. `Triggered` (default and recommended) opens the ProductExplorer GUI when all the selected datasets have been reduced. `Disabled` does not display the ProductExplorer GUI.

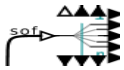
5.2 Workflow Actors

5.2.1 Simple Actors

Simple actors have workflow symbols that consist of a single (rather than multiple) green-blue rectangle. They may also have an icon within the rectangle to aid in their identification. The following actors are simple actors:

-

- The `DataOrganiser` actor.
- 

- The `DataSetChooser` actor (inside a composite actor).
- 

- The `FitsRouter` actor Redirects files according to their categories.
-

- The `ProductRenamer` actor.
-

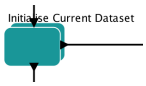
- The `ProductExplorer` actor (inside a composite actor).

Access to the parameters for a simple actor is achieved by right-clicking on the actor and selecting `Configure Actor`. This will open an “Edit parameters” window. Note that the `Product Renamer` actor is a jython script (Java implementation of the Python interpreter) meant to be customised by the user (by double-clicking on it).

5.2.2 Composite Actors

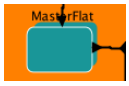
Composite Actors have workflow symbols that consist of multiple-layered green-blue rectangles. They generally do not have a logo within the rectangle. A Composite Actor represents a combination of more Simple or Composite Actors which hides over-complexity from the user in the top-level workflow. In the GIRAFFE workflow, the following actors are composite actors:

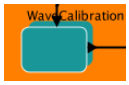
-

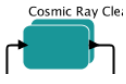
- The `Initialise` actor.
- 

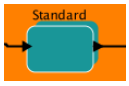
- The `Initialise Current DataSet` actor.
-

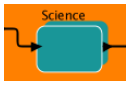
- The `MasterBias` actor.
-

- The `MasterDark` actor.
- 

- The `MasterFlat` actor.
- 

- The `WaveCalibration` actor.
- 

- The `Cosmic Ray Cleaning` actor.
- 

- The `Standard` actor.
- 

- The `Science` actor.
-

- The `Close DataSet` actor.
-

- The `Product Explorer` actor.

Composite Actors may also be expanded for inspection. To do this, right-click on the actor and select `Open Actor`, which will expand the Composite Actor components in a new `Reflex` canvas window. If the Composite Actor corresponds to a pipeline recipe, then the corresponding `RecipeExecuter` actor will be present as a Simple Actor, and its parameters are accessible as for any other Simple Actor. Alternatively you may find further Composite Actors, in which case you need open that/those Composite Actor(s) in order to access the `Recipe Executer`.

5.2.3 Recipe Execution within Composite Actors

The GIRAFFE workflow contains Composite Actors to run pipeline recipes. This is, in the most simple case, due to the `SoF Splitter`/`SoF Accumulator`[13], which allows processing calibration data

[13] SoF stands for Set of Files, which is an ASCII file containing the name (and path) of each input file and its category (e.g. `BIAS`).

from different settings within one given DataSet (e.g. lamp frames taken with different slits/masks). More complex Composite Actors contain several actors (e.g. `Recipe Executer`).

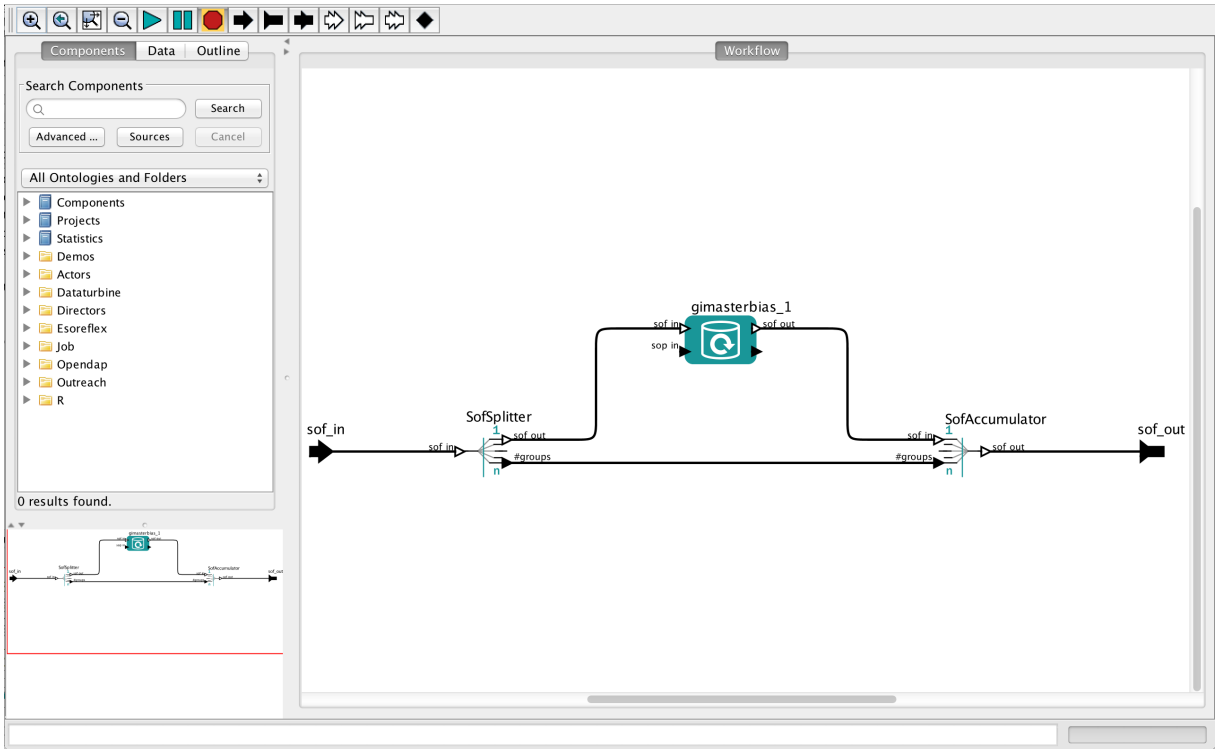


Figure 5.1: *This is the window you get when you choose* `Open Actor` *for the Composite Actor* `masterbias`*. This is the most simple case for a Composite Actor. Using* `Configure Actor` *on* `fors_bias_1` *gives you Fig. 5.2.*

The central elements of any Reflex workflow are the `RecipeExecuter` actors that actually run the recipes. One basic way to embed a `RecipeExecuter` in a workflow is shown in Fig 5.1, which is the most simple version of a Composite Actor. The `RecipeExecuter` is preceded by an `SofSplitter`, and followed by an `SofAccumulator`. The function of the `SofSplitter` is to investigate the incoming SoFs, sort them by "purpose", and create separate SoFs for each purpose. The `RecipeExecuter` then processes each of the SoFs independently (unless they are actually the same files). Finally, the `SofAccumulator` packs all the results into a single output SoF. The direct relation between the `SofSplitter` and `SofAccumulator` is used to communicate the number of different SoFs created by the `SofSplitter`. A workflow will only work as intended if the purpose of all the files a recipe needs as input is identical. The only exception to this rule is that a purpose can also be "default". In this case, the file is included in any output SoF created by the `SoFsplitter` and `SofAccumulator`.

The reason for this scheme is best explained by an example. For a complex DataSet, the `Data Organiser` might have selected a standard star to be processed with the science file. The standard and science files may be associated to two different sets of flats. The `Data Organiser` determines and records this "purpose" for each flat field, and this information is included in the DataSet and each SoF created from this DataSet. The `FitsRouter` directs all raw flat field frames to the

Table 5.1: The GIRAFFE pipeline actors and their contents

actor	recipes	description
MasterBias	`gimasterbias`	Creates a master bias image from a set of raw biases
MasterDark	`gimasterdark`	Creates a master dark image from a set of raw dark frames
MasterFlat	`gimasterflat`	Create the fiber master flat field and the localization mask
WaveCalibration	`giwavecalibration`	Compute dispersion solution from an arc-lamp spectrum
Cosmic Ray Cleaning	*python scripts*	Create additional versions of the science frames with Cosmic Rays cleaned *if* CRC python packages are installed
Standard	`gistandard`	Process a spectro-photometric standard star observation and compute the instrument response curve
Science	`giscience`	Process a science observation

`MasterFlat` Composite Actor. The `SofSplitter` then creates SoFs, one for the science frame and one for the standard star frame, for the flat fields to be used for the science and standard star frames. The `gimasterflat` recipe creates one master flat field (and other products) for each SoF, and the `SofAccumulator` then creates a SoF that contains all the products.

A `RecipeExecuter` actor is used in the workflow to run a single GIRAFFE pipeline recipe (e.g: in the `MasterBias` actor the recipe `gimasterbias` is executed). In order to configure the `RecipeExecuter`s, one has to first use `Open Actor` to get to the level of the recipe executors (see Fig. 5.1).

In Figure 5.2 we show the 'Edit parameters' window for a typical `RecipeExecuter` actor, which can be displayed by right-clicking on the actor and selecting `Configure Actor`. In the following we describe in more detail the function of some of the parameters for a `RecipeExecuter` actor:

- The "recipe" parameter states the GIRAFFE pipeline recipe which will be executed.
- The "mode" parameter has a pull-down menu allowing the user to specify the execution mode of the actor. The available options are:
  - `Run`: The pipeline recipe will be executed, possibly in Lazy mode (see Section 5.2.5). This option is the default option.
  - `Skip`: The pipeline recipe is not executed, and the actor inputs are passed to the actor outputs.
  - `Disabled`: The pipeline recipe is not executed, and the actor inputs are not passed to the actor outputs.
- The "Lazy Mode" parameter has a tick-box (selected by default) which indicates whether the `RecipeExecuter` actor will run in Lazy mode or not. A full description of Lazy mode is provided in Sect. 5.2.5.

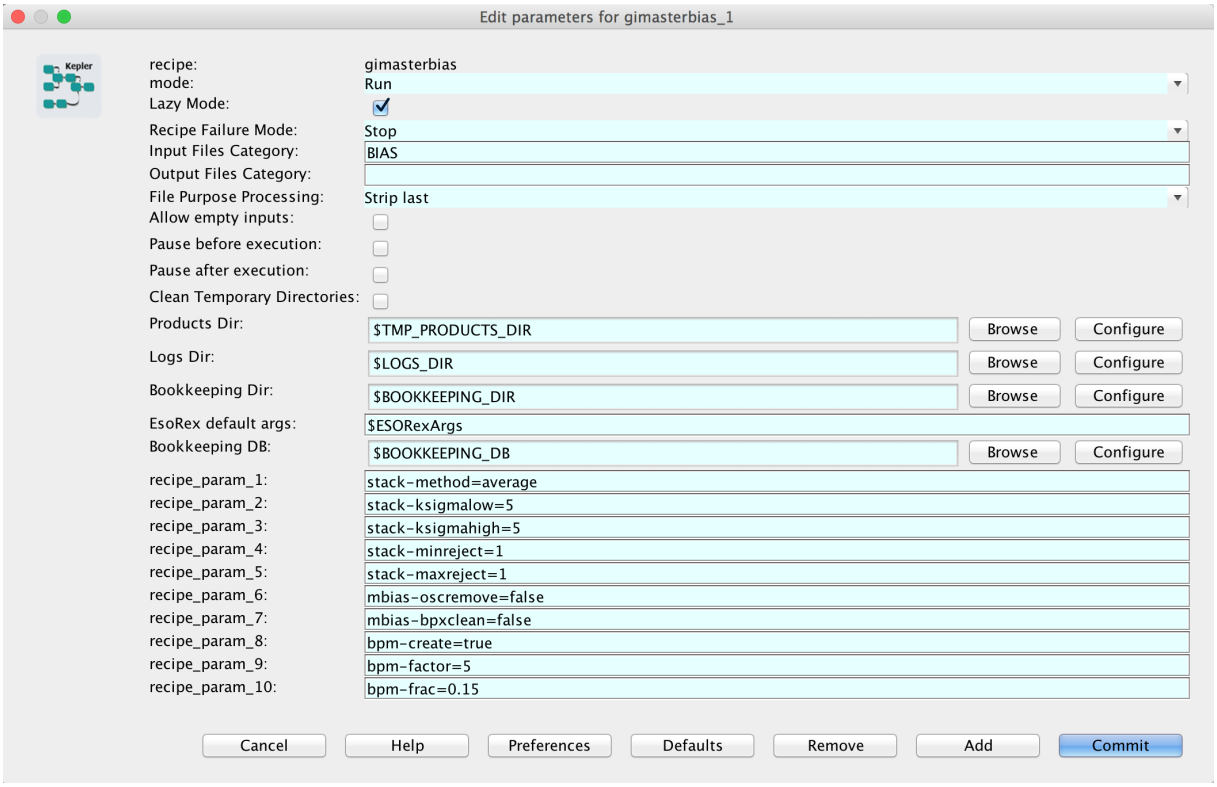


Figure 5.2: *The 'Edit parameters' window for a typical* `RecipeExecuter` *actor, the* `gimasterbias_1` *actor which runs the* `gimasterbias` *pipeline recipe.*

- The "Recipe Failure Mode" parameter has a pull-down menu allowing the user to specify the behaviour of the actor if the pipeline recipe fails. The available options are:
  - `Stop`: The actor issues an error message and the workflow stops.
  - `Continue`: The actor creates an empty output and the workflow continues.
  - `Ask`: The actor displays a pop-up window and asks the user whether he/she wants to continue or stop the workflow. This option is the default option.
- The set of parameters which start with 'recipe param' and end with a number or a string correspond to the parameters of the relevant GIRAFFE pipeline recipe. By default in the `RecipeExecuter` actor, the pipeline recipe parameters are either set to their pipeline default values, or to the value 'PORT', which means that the parameter can be adjusted in the corresponding Interacrive GUI.

5.2.4 Adjusting any recipe parameter

If you need to change the value of a parameter that is not accessible via the GUI, you need to stop the workflow, adjust the parameter here in the relevant 'Edit Parameters' window, and then restart the

workflow.[14]

The description of the remainder of the `RecipeExecuter` actor parameters are outside the scope of this tutorial, and the interested user is referred to the Reflex User Manual for further details ([8]). Any changes that you make in the "Edit parameters" window may be saved in the workflow by clicking the `Commit` button when you have finished. **Do NOT simply type 'Enter' as this will cause quotes to be added to the config file and cause the worflow to crash.**

5.2.5 Lazy Mode

By default, all `RecipeExecuter` actors in a pipeline workflow are "Lazy Mode" enabled. This means that when the workflow attempts to execute such an actor, the actor will check whether the relevant pipeline recipe has already been executed with the same input files and with the same recipe parameters. If this is the case, then the actor will not execute the pipeline recipe, and instead it will simply broadcast the previously generated products to the output port. The purpose of the Lazy Mode is therefore to minimise any reprocessing of data by avoiding data re-reduction where it is not necessary.

One should note that the actor's Lazy Mode depends on the contents of the directory specified by the parameter `BOOKKEEPING_DIR` and the relevant FITS file checksums. Any modification to the directory contents and/or the file checksums will cause the corresponding actor to run the pipeline recipe again when executed, thereby re-reducing the input data.

The re-reduction of data at each execution may sometimes be desirable. To force a re-reduction of data for any single `RecipeExecuter` actor in the workflow, right-click the actor, select `Configure Actor`, and uncheck the Lazy mode parameter tick-box in the "Edit parameters" window that is displayed. For many workflows the `RecipeExecuter` actors are actually found inside the composite actors in the top level workflow. To access such embedded `RecipeExecuter` actors you will first need to open the sub-workflow by right-clicking on the composite actor and then selecting `Open Actor`.

To force the re-reduction of all data in a workflow (i.e. to disable Lazy mode for the whole workflow), you must uncheck the Lazy mode for every single `RecipeExecuter` actor in the entire workflow. It is also possible to change the name of the bookkeeping directory, instead of modifying any of the Lazy mode parameters. This will also force a re-reduction of the given dataset(s). A new reduction will start (with the lazy mode still enabled), but the results of previous reduction will not be reused. Alternatively, if there is no need to keep any of the previously reduced data, one can simply set the `EraseDirs` parameter under the "Global Parameters" area of the workflow canvas to `true`. This will then remove all previous results that are stored in the bookkeeping, temporary, and log directories before processing the input data, in effect, starting a new clean data reduction and re-processing every input dataset. *Note: The option* `EraseDirs` = `true` *does not work in* `esoreflex` *version 2.9.x and makes the workflow to crash.*

[14] If you think a particular parameter should be adjustable via the GUI, please contact *https://support.eso.org* to request that it is include in the GUI in a future release.

6 Reducing your own data

In this section we describe how to reduce your own data set.

First, we suggest the reader to familiarize with the workflow by reducing the demo dataset first (Section 3), but it is not a requirement.

6.1 The `esoreflex` command

We list here some options associated to the `esoreflex` command. We recommend to try them to familiarize with the system. In the following, we assume the `esoreflex` executable is in your path; if not you have to provide the full path `<install_dir>/bin/esoreflex`

To see the available options of the `esoreflex` command type:

```
esoreflex -h
```

The output is the following.

```
-h | -help              print this help message and exit.
-v | -version           show installed Reflex version and pipelines and exit.
-l | -list-workflows    list available installed workflows and from
                        ~/KeplerData/workflows.
-n | -non-interactive   enable non-interactive features.
-e | -explore           run only the Product Explorer in this workflow
-p <workflow> | -list-parameters <workflow>
                        lists the available parameters for the given
                        workflow.
-config <file>          allows to specify a custom esoreflex.rc configuration
                        file.
-create-config <file>   if <file> is TRUE then a new configuration file is
                        created in ~/.esoreflex/esoreflex.rc. Alternatively
                        a configuration file name can be given to write to.
                        Any existing file is backed up to a file with a '.bak'
                        extension, or '.bakN' where N is an integer.
-debug                  prints the environment and actual Reflex launch
                        command used.
```

6.2 Launching the workflow

We list here the recommended way to reduce your own datasets. Steps 1 and 2 are optional and one can start from step 3.

1. Type: `esoreflex -n <parameters>` GIRAFFE to launch the workflow non interactively and reduce all the datasets with default parameters.

   `<parameters>` allows you to specify the workflow parameters, such as the location of your raw data and the final destination of the products.

   For example, type (in a single command line):

   ```
   esoreflex -n
     -RAW_DATA_DIR /home/user/my_raw_data
     -ROOT_DATA_DIR /home/user/my_reduction
     -END_PRODUCTS_DIR $ROOT_DATA_DIR/reflex_end_products
     giraf
   ```

   to reduce the complete datasets that are present in the directory `/home/user/my_raw_data` and that were not reduced before. Final products will be saved in `/home/user/my_reduction/reflex_end_products`, while book keeping, temporary products, and logs will be saved in sub-directories of `/home/user/my_reduction/`. If the reduction of a dataset fails, the reduction continues to the next dataset. It can take some time, depending on the number of datasets present in the input directory. For a full list of workflow parameters type `esoreflex -p` GIRAFFE. Note that this command lists only the parameters, but does not launch the workflow.

   Once the reduction is completed, one can proceed with optimizing the results with the next steps.

2. Type:

   ```
   esoreflex -e giraf
   ```

   to launch the Product Explorer. The Product Explorer allows you to inspect the data products already reduced by the GIRAFFE `esoreflex` workflow. Only products associated with the workflow default bookkeeping database are shown. To visualize products associated to given bookkeeping database, pass the full path via the `BOOKKEEPING_DB` parameter:

   ```
   esoreflex -e BOOKKEEPING_DB <database_path> giraf
   ```

   to point the product explorer to a given `<database_path>`, e.g., `/home/username/reflex/reflex_bookkeeping/test.db`

   The Product Explorer allows you to inspect the products while the reduction is running. Press the button `Refresh` to update the content of the Product Explorer. This step can be launched in parallel to step 1.

   A full description of the Product Explorer will be given in Section 6.3.3

3. Type:

   ```
   esoreflex giraf &
   ```

   to launch the GIRAFFE `esoreflex` workflow. The GIRAFFE workflow window will appear (Fig. 3.1). Please configure the set-up directories `ROOT_DATA_DIR`, `RAW_DATA_DIR`, and other workflow parameters as needed. Just double-click on them, edit the content, and press `OK`. Remember to specify the same `<database_path>` as for the Product Explorer, if it has been opened at step #2, to synchronize the two processes.

4. (Recommended, but not mandatory) On the main `esoreflex` menu set `Tools` –> `Animate at Runtime` to 1 in order to highlight in red active actors during execution.

5. Press the button to start the workflow. First, the workflow will highlight and execute the `Initialise` actor, which among other things will clear any previous reductions if required by the user (see Section 5.1). Secondly, if set, the workflow will open the Product Explorer, allowing the user to inspect previously reduced datasets (see Section 6.3.3 for how to configure this option).

6.3 Workflow Steps

6.3.1 Data Organisation And Selection

The `DataOrganiser` (DO) is the first crucial component of a Reflex workflow. The DO takes as input `RAW_DATA_DIR` and `CALIB_DATA_DIR` and it detects, classifies, and organises the files in these directories and any subdirectories. The output of the DO is a list of "DataSets". A DataSet is a special Set of Files (SoF). A DataSet contains one or several science (or calibration) files that should be processed together, and all files needed to process these data. This includes any calibration files, and in turn files that are needed to process these calibrations. Note that different DataSets might overlap, i.e. some files might be included in more than one DataSet (e.g., common calibration files).

A DataSet lists three different pieces of information for each of its files, namely 1) the file name (including the path), 2) the file category, and 3) a string that is called the "purpose" of the file. The DO uses the OCA[15] rules to find the files to include in a DataSet, as well as their categories and purposes. The file category identifies different types of files, and it is derived by information in the header of the file itself. A category could for example be `RAW_CALIBRATION_1`, `RAW_CALIBRATION_2` or `RAW_SCIENCE`, depending on the instrument. The purpose of a file identifies the reason why a file is included in a DataSet. The syntax is `action_1/action_2/action_3/ ... /action_n`, where each `action_i` describes an intended processing step for this file (for example, creation of a `MASTER_CALIBRATION_1` or a `MASTER_CALIBRATION_2`). The actions are defined in the OCA rules and contain the recipe together with all file categories required to execute it (and predicted products in case of calibration data). For example, a workflow might include two actions `action_1` and `action_2`. The former creates `MASTER_ CALIBRATION_1` from `RAW_CALIBRATION_1`, and the later creates a `MASTER_CALIBRATION_2` from `RAW_CALIBRATION_2`. The `action_2` action needs `RAW_CALIBRATION_2` frames and the `MASTER_ CALIBRATION_1` as input. In this case, these `RAW_CALIBRATION_1` files will have the purpose `action_ 1/action_2`. The same DataSet might also include `RAW_CALIBRATION_1` with a different purpose; irrespective of their purpose the file category for all these biases will be `RAW_CALIBRATION_1`.

[15] OCA stands for OrganisationClassificationAssociation and refers to rules, which allow to classify the raw data according to the contents of the header keywords, organise them in appropriate groups for processing, and associate the required calibration data for processing. They can be found in the directory `<install_dir>/share/esopipes/<pipeline-version>/reflex/`, carrying the extension `.oca`. The variable `<install_dir>` depends on the operative system and installation procedure. For installation through `rpm`: `<install_dir>=/usr`; for installation through macport `<install_dir>=/opt/local`; for installation through the installation script install_esoreflex it depends on the path specified during installation, e.g. `<install_dir>=<specified_path>/install`

The Datasets created via the `DataOrganiser` will be displayed in the `DataSet Chooser`. Here the users have the possibility to inspect the various datasets and decide which one to reduce. By default, DataSets that have not been reduced before are highlighted for reduction. Click either `Continue` in order to continue with the workflow reduction, or `Stop` in order to stop the workflow. A full description of the `DataSet Chooser` is presented in Section 6.3.2.

Once the `Continue` is pressed, the workflow starts to reduce the first selected DataSet. Files are broadcasted according to their purpose to the relevant actors for processing.

The categories and purposes of raw files are set by the DO, whereas the categories and purpose of products generated by recipes are set by the `RecipeExecuter`. The file categories are used by the `FitsRouter` to send files to particular processing steps or branches of the workflow (see below). The purpose is used by the `SofSplitter` and `SofAccumulator` to generate input SoFs for the `RecipeExecuter`. The `SofSplitter` and `SofAccumulator` accept several SoFs as simultaneous input. The `SofAccumulator` creates a single output SoF from the inputs, whereas the `SofSplitter` creates a separate output SoF for each purpose.

6.3.2 DataSetChooser

The `DataSetChooser` displays the DataSets available in the “Select Data Sets” window, activating vertical and horizontal scroll bars if necessary (Fig. 3.2).

Some properties of the DataSets are displayed: the name, the number of files, a flag indicating if it has been successfully reduced (a green OK), if the reduction attempts have failed or were aborted (a red FAILED), or if it is a new dataset (a black "-"). The column "Descriptions" lists user-provided descriptions (see below), other columns indicate the instrument set-up and a link to the night log.

Sometimes you will want to reduce a subset of these DataSets rather than all DataSets, and for this you may individually select (or de-select) DataSets for processing using the tick boxes in the first column, and the buttons `Deselect All` and `Select Complete` at the bottom, or configure the “Filter” field at the bottom left. Available filter options are: "New" (datasets not previously reduced will be selected), "Reduced" (datasets previously reduced will be selected), "All" (all datasets will be selected), and "Failed" (dataset with a failed or aborted reduction will be selected).

You may also highlight a single DataSet in blue by clicking on the relevant line. If you subsequently click on `Inspect Highlighted`, then a “Select Frames” window will appear that lists the set of files that make up the highlighted DataSet including the full filename[16], the file category (derived from the FITS header), and a selection tick box in the right column. The tick boxes allow you to edit the set of files in the DataSet which is useful if it is known that a certain calibration frame is of poor quality (e.g: a poor raw flat-field frame). The list of files in the DataSet may also be saved to disk as an `ASCII` file by clicking on `Save As` and using the file browser that appears.

By clicking on the line corresponding to a particular file in the “Select Frames” window, the file will be highlighted in blue, and the file FITS header will be displayed in the text box on the right, allowing a quick inspection of useful header keywords. If you then click on `Inspect`, the workflow will open the file in the selected FITS viewer application defined by the workflow parameter `FITS_VIEWER`.

[16] keep the mouse pointer on the file name to visualize the full path name.

To exit from the “Select Frames” window, click `Continue`.

To add a description of the reduction, press the button `...` associated with the field "Add description to the current execution of the workflow" at the bottom right of the Select Dataset Window; a pop up window will appear. Enter the desired description (e.g. "My first reduction attempt") and then press `OK`. In this way, all the datasets reduced in this execution, will be flagged with the input description. Description flags can be visualized in the `SelectFrames` window and in the `ProductExplorer`, and they can be used to identify different reduction strategies.

To exit from the “Select DataSets” window, click either `Continue` in order to continue with the workflow reduction, or `Stop` in order to stop the workflow.

6.3.3 The ProductExplorer

The ProductExplorer is an interactive component in the `esoreflex` workflow whose main purpose is to list the final products with the associated reduction tree for each dataset and for each reduction attempt (see Fig. 3.11).

Configuring the ProductExplorer

You can configure the ProductExplorer GUI to appear after or before the data reduction. In the latter case you can inspect products as reduction goes on.

1. To display the ProductExplorer GUI at the end of the datareduction:

   - Click on the global parameter “ProductExplorerMode” before starting the data reduction. A configuration window will appear allowing you to set the execution mode of the Product Explorer. Valid options are:
     - "Triggered" (default). This option opens the ProductExplorer GUI when all the selected datasets have been reduced.
     - "Enabled". This option opens the ProductExplorer GUI at the end of the reduction of each individual dataset.
     - “Disable”. This option does not display the ProductExplorer GUI.
   - Press the button to start the workflow.

2. To display the ProductExplorer GUI “before” starting the data reduction:

   - double click on the composite Actor "Inspect previously reduced data". A configuration window will appear. Set to "Yes" the field "Inspect previously reduced data (Yes/No)". Modify the field "Continue reduction after having inspected the previously reduced data? (Continue/Stop/Ask)". "Continue" will continue the workflow and trigger the DataOrganizer. "Stop" will stop the workflow; "Ask" will prompt another window deferring the decision whether continuing or not the reduction after having closed the Product Explorer.

- Press the button to start the workflow. Now the ProductExplorer GUI will appear before starting the data organization and reduction.

Exploring the data reduction products

The left window of the ProductExplorer GUI shows the executions for all the datasets (see Fig. 3.11). Once you click on a dataset, you get the list of reduction attemps. Green and red flags identify successfull or unsucessfull reductions. Each reduction is linked to the “Description” tag assigned in the “Select Dataset” window.

1. To identify the desired reduction run via the “Description” tag, proceed as follows:

   - Click on the symbol at the left of the dataset name. The full list of reduction attempts for that dataset will be listed. The column Exec indicates if the reduction was succesful (green flag: "OK") or not (red flag: "Failed").
   - Click on the entries in the field "Description" to visualize the description you have entered associated to that dataset on the Select Dataset window when reducing the data.
   - Identify the desired reduction run. All the products are listed in the central window, and they are organized following the data reduction cascade.

You can narrow down the range of datasets to search by configuring the field "Show" at the top-left side of the ProductExplorer (options are: "All", "Successful", "Unsuccessful"), and specifying the time range (Last, all, From-to).

2. To inspect the desired file, proceed as follows:

   - Navigate through the data reduction cascade in the ProductExplorer by clicking on the files.
   - Select the file to be inspected and click with the mouse right-hand button. The available options are:
     - Options available always:
       * Copy full path. It copies the full name of the file onto the clipboard. Shift+Ctr+v to past it into a terminal.
       * Inspect Generic. It opens the file with the fits viewer selected in the main workflow canvas.
       * Inspect with. It opens the file with an executable that can be specified (you have to provide the full path to the executable).
     - Options available for files in the `TMP_PRODUCTS_DIR` directory only:
       * command line. Copy of the environment configuration and recipe call used to generate that file.
       * Xterm. It opens an Xterm at the directory containing the file.

– Options available for products associated to interactive windows only:

  * Display pipeline results. It opens the interactive windows associated to the recipe call that generated the file. Note that this is for visualization purposes only; the recipe parameters cannot be changed and the recipe cannot be re-run from this window.

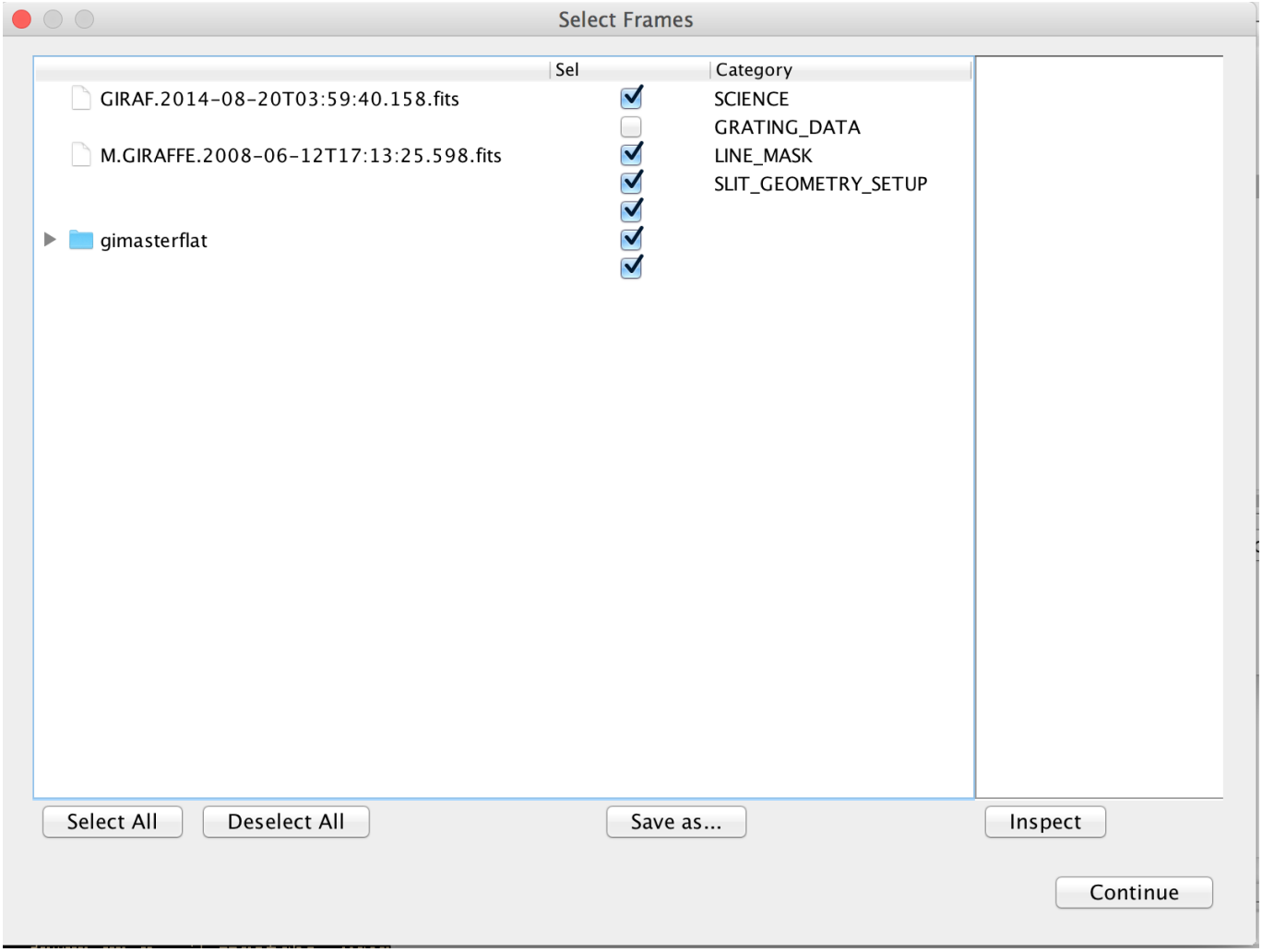


Figure 6.1: *The "Select Frames" window with a single file from the current Data Set highlighted in blue, and the corresponding FITS header displayed in the text box on the right. Hidden partially behind the "Select Frames" window is the "Select DataSets" window with the currently selected DataSet highlighted. On some systems some of the information 'disappears' after being displayed for a fraction of a second (as in figure 6.1). To re-display all the information, just click the* `Select All` *button, you may then want to de-select again the frames that were de-selected by default, e.g. the* `GRATING_DATA` *frame in the example above..*

6.3.4 Creation of Master Calibration Files

In this step of the workflow, the following GIRRAFE recipes are executed in the order listed below. Please refer to the GIRAFFE pipeline user manual ([4]: Sections 9 and 10) for the details of each recipe and the algorithms employed:

1. The `MasterBias` actor will execute the GIRAFFE pipeline recipe `gimasterbias` in order to create a combined master bias frame from the set of raw bias frames

2. The `MasterDark` actor will execute the GIRAFFE pipeline recipe `gimasterdark` in order to create a combined master dark frame from the set of raw dark frames

3. The `MasterFlat` actor will execute the GIRAFFE pipeline recipe `gimasterflat` in order detect and trace each of the fibres and to create an extracted and normalised flat-field frame from the set of raw flat frames or raw naysmyth-flat frames

4. The `Wavelength Calibration` actor will execute the GIRAFFE pipeline recipe `giwavecalibration` in order to compute the dispersion solution, and, optionally, a slit geometry table for the fibre setup in use from the raw wavelength calibration frame

5. The `Cosmic Ray Cleaning` creates alternative versions of the input RAW science and standard star files corrected for cosmic rays *if* the required Cosmic Ray Cleaning python packages have been installed on your system and made available to Reflex. This actor will also create 'medianed' versions if the grouped workflow is used and the dataset include 3 or more science/standard star input files – note when grouped is used the pipleine combines the multiple frames with a simple mean. The Cosmic ray Cleaning actor typically executes immediately after the MAsterBIAS (or MatserDARK) actor.

6.3.5 Response Computation

The `Standard` actor will execute the GIRAFFE pipeline recipe `gistandard` in order to compute the response function of the system, **only applicable to IFU and ARGUS modes**. Please also note that the workflow does not (yet) apply this response function in any way. This needs to be done 'manually' outside the Reflex workflow[17]

6.3.6 Science Reduction

The `Science` actor will execute the GIRAFFE pipeline recipe `giscience` to bias, flat-field, fibre-to-fibre correct and extract wavelength calibrated spectra. Please refer to the GIRAFFE pipeline user manual ([4]: Sections 9 and 10) for the details of this recipe and the extraction algorithms employed.

[17] Unless of course you develop your own version of the worflow to implement response function calibration...

6.3.7 Output Organisation

After having processed the input data for a DataSet, the workflow highlights and executes the `Product Renamer` actor, which, by default, will copy the defined final products of the `Science` actor to the directory specified by `END_PRODUCTS_DIR` and rename them with names derived from the values of certain FITS header keywords. Specifically, final products are organised into subdirectories corressponding the the Cosmic Ray Cleaning method used, i.e. `CMETHOD=<CMETHOD>` The files are renamed by default with names of the form `<HIERARCH.ESO.OBS.NAME>_<HIERARCH.ESO.PRO.CATG>_<CMETHOD>.fits`, with `<CMETHOD>`, `<HIERARCH.ESO.OBS.NAME>` and `<HIERARCH.ESO.PRO.CATG>` representing the values of the corresponding FITS header keywords, and `<CMETHOD>` usually has one of the values `NONE`, `ASTROSCRAPPY` or `PYCOSMIC`. These names are fully configurable by right-clicking on the `Product Renamer` actor, selecting `Configure Actor`, and then editing the string as appropriate.

For Medusa data the final products that are copied and renamed are:

- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_EXTERRORS_<CMETHOD>.fits` Statistical error (standard deviation) of the extracted science spectra
- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_EXTPIXELS_<CMETHOD>.fits` Number of pixels contributing to each extracted wave- length bin of the science spectra (standard and Horne extraction only)
- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_EXTSPECTRA_<CMETHOD>.fits` The extracted fluxes in ADU of the science spectra for each wavelength bin. The fluxes are extracted from the reduced science frame. The spectra may also be flat field corrected.
- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_EXTTRACES_<CMETHOD>.fits` The centroid position of the extracted fluxes for each wavelength bin.
- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_RBNERRORS_<CMETHOD>.fits` Statistical error (standard deviation) of the rebinned science spectra
- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_RBNSPECTRA_<CMETHOD>.fits` Rebinned science spectra
- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_REDUCED_<CMETHOD>.fits` Bias subtracted average of all input raw science observations

For IFU data the final products that are copied and renamed are:

- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_CUBE_ERRORS_<CMETHOD>.fits` Data cube of the rebinned spectra errors
- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_CUBE_SPECTRA_<CMETHOD>.fits` Data cube of the rebinned spectra
- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_EXTERRORS_<CMETHOD>.fits` Statistical error (standard deviation) of the extracted science spectra

- <HIERARCH.ESO.OBS.NAME>_SCIENCE_EXTPIXELS_<CMETHOD>.fits Number of pixels contributing to each extracted wave- length bin of the science spectra (standard and Horne extraction only)
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_EXTSPECTRA_<CMETHOD>.fits The extracted fluxes in ADU of the science spectra for each wavelength bin. The fluxes are extracted from the reduced science frame. The spectra may also be flat field corrected.
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_EXTTRACES_<CMETHOD>.fits The centroid position of the extracted fluxes for each wavelength bin.
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_RBNERRORS_<CMETHOD>.fits Statistical error (standard deviation) of the rebinned science spectra
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_RBNSPECTRA_<CMETHOD>.fits Rebinned science spectra
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_RCERRORS_<CMETHOD>.fits Statistical error (standard deviation) of the recon- structed field of view
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_RCSPECTRA_<CMETHOD>.fits Reconstructed field of view
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_REDUCED_<CMETHOD>.fits Bias subtracted average of all input raw science observations

For ARGUS data the final products that are copied and renamed are:

- <HIERARCH.ESO.OBS.NAME>_SCIENCE_CUBE_ERRORS_<CMETHOD>.fits Data cube of the rebinned spectra errors
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_CUBE_SPECTRA_<CMETHOD>.fits Data cube of the rebinned spectra
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_EXTERRORS_<CMETHOD>.fits Statistical error (standard deviation) of the extracted science spectra
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_EXTPIXELS_<CMETHOD>.fits Number of pixels contributing to each extracted wave- length bin of the science spectra (standard and Horne extraction only)
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_EXTSPECTRA_<CMETHOD>.fits The extracted fluxes in ADU of the science spectra for each wavelength bin. The fluxes are extracted from the reduced science frame. The spectra may also be flat field corrected.
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_EXTTRACES_<CMETHOD>.fits The centroid position of the extracted fluxes for each wavelength bin.
- <HIERARCH.ESO.OBS.NAME>_SCIENCE_RBNERRORS_<CMETHOD>.fits Statistical error (standard deviation) of the rebinned science spectra

- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_RBNSPECTRA_<CMETHOD>.fits` Rebinned science spectra
- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_RCERRORS_<CMETHOD>.fits` Statistical error (standard deviation) of the recon- structed field of view
- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_RCSPECTRA_<CMETHOD>.fits` Reconstructed field of view
- `<HIERARCH.ESO.OBS.NAME>_SCIENCE_REDUCED_<CMETHOD>.fits` Bias subtracted average of all input raw science observations

The following actors in this step of the workflow are concerned with the termination of the data flow for the current DataSet and will highlight briefly as they are executed.

Finally, once all datasets have been processed, the `Provenance Explorer` window will appear as shown in Fig. 3.11 with a list of datasets on the left menu. By unfolding the menu under each dataset, all the renamed products appear, and if one is interested in the files, including all intermediate steps, that are used to produce that final product, just click on it and a dependency tree will show the whole reduction chain.

7 Frequently Asked Questions

- **The error window fills the whole screen - how can I get to the `Continue`/`Stop` buttons?**

  Press the Alt key together with your left mouse button to move the window upwards and to the left. At the bottom the `Continue`/`Stop` buttons will be visible. This bug is known but could not yet be fixed.

- **I tried to `Open` (or `Configure`) an `Actor` while the workflow is running and now it does not react any more. What should I do?**

  This is a limitation of the underlying Kepler engine. The only way out is to kill the workflow externally. If you want to change anything while a workflow is running you first need to pause it.

- **After a successful reduction of a data set, I changed this data set in some way (e.g. modified or removed some files, or changed the rules of the Data Organizer). When I restart Reflex, the Data Set Chooser correctly displays my new data set, but marks it as "reduced ok", even though it was never reduced before. What does this mean?**

  The labels in the column "Reduced" of the Data Set Chooser mark each dataset with "OK", "Failed" or "-". These labels indicate whether a data set has previously successfully been reduced at least once, all previous reductions failed, or a reduction has never been tried respectively. Data sets are identified by their name, which is derived from the first science file within the data set. As long as the data set name is preserved (i.e. the first science file in a data set has not changed), the Data Organizer will consider it to be the same data set. The Data Organizer recognizes any previous reductions of data sets it considers to be the same as the current one, and labels the current data set with "OK" if any of them was successful, even if the previously reduced data set differs from the current one.

  Note that the Product Explorer will list all the previous reductions of a particular data set only at the end of the reduction. This list might include successful and/or unsuccessful reduction runs with different parameters, or in your case with different input files. The important fact is that these are all reductions of data sets with the same first raw science file. By browsing through all reductions of a particular raw science file, the users can choose the one they want to use.

- **Where are my intermediate pipeline products?** Intermediate pipeline products are stored in the directory `<TMP_PRODUCTS_DIR>` (defined on the workflow canvas, under Setup Directories) and organised further in directories by pipeline recipe.

- **Can I use different sets of bias frames to calibrate my flat frames and science data?** Yes. In fact this is what is currently implemented in the workflow(s). Each file in a DataSet has a purpose attached to it ([8]). It is this purpose that is used by the workflow to send the correct set of bias frames to the recipes for flat frame combination and science frame reduction, which may or may not be the same set of bias frames in each case.

- **Can I run `Reflex` from the command line?** Yes, use the command:

```
esoreflex -n <workflow_path>/<workflow>.xml
```

The -n option will set all the different options for Kepler and the workflows to avoid opening any GUI elements (including pipeline interactive windows).

It is possible to specify workflow variables (those that appear in the workflow canvas) in the command line. For instance, the raw data directory can be set with this command:

```
esoreflex -n -RAW_DATA_DIR <raw_data_path> \
              <workflow_path>/<workflow>.xml
```

You can see all the command line options with the command `esoreflex -h`.

Note that this mode is not fully supported, and the user should be aware that the path to the workflow must be absolute and even if no GUI elements are shown, it still requires a connection to the window manager.

- **How can I add new actors to an existing workflow?** You can drag and drop the actors in the menu on the left of the `Reflex` canvas. Under `Eso-reflex -> Workflow` you may find all the actors relevant for pipeline workflows, with the exception of the recipe executer. This actor must be manually instantiated using `Tools -> Instantiate Component`. Fill in the "Class name" field with `org.eso.RecipeExecuter` and in the pop-up window choose the required recipe from the pull-down menu. To connect the ports of the actor, click on the source port, holding down the left mouse button, and release the mouse button over the destination port. Please consult the Reflex User Manual ([8]) for more information.

- **How can I broadcast a result to different subsequent actors?** If the output port is a multi-port (filled in white), then you may have several relations from the port. However, if the port is a single port (filled in black), then you may use the black diamond from the toolbar. Make a relation from the output port to the diamond. Then make relations from the input ports to the diamond. Please note that you cannot click to start a relation from the diamond itself. Please consult the Reflex User Manual ([8]) for more information.

- **How can I manually run the recipes executed by Reflex?** If a user wants to re-run a recipe on the command line he/she has to go to the appropriate reflex_book_keeping directory, which is generally reflex_book_keeping/<workflow>/<recipe_name>_<number> There, subdirectories exist with the time stamp of the recipe execution (e.g. 2013-01-25T12:33:53.926/). If the user wants to re-execute the most recent processing he/she should go to the `latest` directory and then execute the script `cmdline.sh`. Alternatively, to use a customized `esorex` command the user can execute

  ```
  ESOREX_CONFIG="INSTALL_DIR/etc/esorex.rc"
  PATH_TO/esorex --recipe-config=<recipe>.rc <recipe> data.sof
  ```

  where INSTALL_DIR is the directory where Reflex and the pipelines were installed.

  If a user wants to re-execute on the command line a recipe that used a specific raw frame, the way to find the proper data.sof in the bookkeeping directory is via `grep <raw_file> */data.sof`. Afterwards the procedure is the same as before.

If a recipe is re-executed with the command explained above, the products will appear in the directory from which the recipe is called, and not in the reflex_tmp_products or reflex_end_products directory, and they will not be renamed. This does not happen if you use the `cmdline.sh` script.

- **Can I reuse the bookkeeping directory created by previous versions of the pipeline?**

  In general no. In principle, it could be reused if no major changes were made to the pipeline. However there are situations in which a previously created bookkeeping directory will cause problems due to pipeline versions incompatibility. This is especially true if the parameters of the pipeline recipes have changed. In that case, please remove the bookkeeping directory completely.

- **How to insert negative values into a textbox?**

  Due to a bug in wxPython, the GUI might appear to freeze when attempting to enter a negative number in a parameter's value textbox. This can be worked around by navigating away to a different control in the GUI with a mouse click, and then navigating back to the original textbox. Once focus is back on the original textbox the contents should be selected and it should be possible to replace it with a valid value, by typing it in and pressing the enter key.

- **I've updated my Reflex installation and when I run esoreflex the process aborts. How can I fix this problem?**

  As indicated in Section 2, in case of major or minor (affecting the first two digit numbers) Reflex upgrades, the user should erase the `$HOME/KeplerData`, `$HOME/.kepler` directories if present, to prevent possible aborts (i.e. a hard crash) of the esoreflex process.

- **How can include my analysis scripts and algorithms into the workflow?**

  EsoReflex is capable of executing any user-provided script, if properly interfaced. The most convenient way to do it is through the Python actor. Please consult the tutorial on how to insert Python scripts into a workflow available here: www.eso.org/sci/data-processing/Python_and_esoreflex.pdf

8 Troubleshooting

1. **The `MasterFlat` actor/`gimasterflat` recipe fails and the workflow crashes because it is unable to locate the correct number of fibres**

   By default the recipe assumes that all of the fibres specified in the `SLIT_GEOMETRY_MASTER` or `SLIT_GEOMETRY_SETUP` calibrations are present. If one or more are not, present, or for whatever reason the pipeline fails to successfully locate one or more of those fibres then the recipe will crash. In order to successfully reduce such data you will need to correctly specify the geometry via the `fiber-splist` parameter in the `gimasterflat_1` actor configuration, see the quickstart guide for instructions to adjust the parameters of a recipe.

   The first indication will be the display of a rediculously wide error message window, see figure 8.1. Mouse-click-dragging it to the left, eventually you will discover the `Stop` and `Continue` buttons, see figure 8.2. Click the `Continue` button and then the MastetFlat GUI interactive window will open (figure 8.3, to display the products (if any) that the recipe did manage to create before crashing and use the Log file window accessed via the `Display Log` button on the right to try to debug what went wrong, and how to set the recipe parameters in order to successfully reduce the dataset.

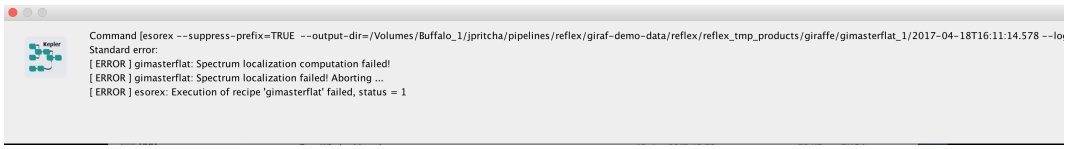


Figure 8.1: *If the MasterFlat recipe fails, a rediculously wide error message window like the one above will popup, by mouse-click-dragging it to the left, eventually you will discover the* `Stop` *and* `Continue` *buttons, see figure 8.2.*

Figure 8.2: *If the MasterFlat recipe fails, a rediculously wide error message window like the one above will popup, once mouse-click-dragged to the left you will discover the* `Stop` *and* `Continue` *buttons. Click the* `Continue` *button and the Interactive MasterFLat GUI window will open, see figure 8.3.*

2. **The "Select DataSets" window displays my DataSets, but some/all of them are greyed out. What is going on?**

   If a DataSet in the "Select DataSets" window is greyed out, then it means that the DataSet which was constructed is missing some key calibration(s) (i.e. the DataSet is incomplete). To find out what calibration(s) are missing from a greyed out DataSet, click on the DataSet in question to highlight it, and then click on the button `Inspect Highlighted`. The "Select Frames" window that appears will report the category of the calibration products that are missing (e.g.

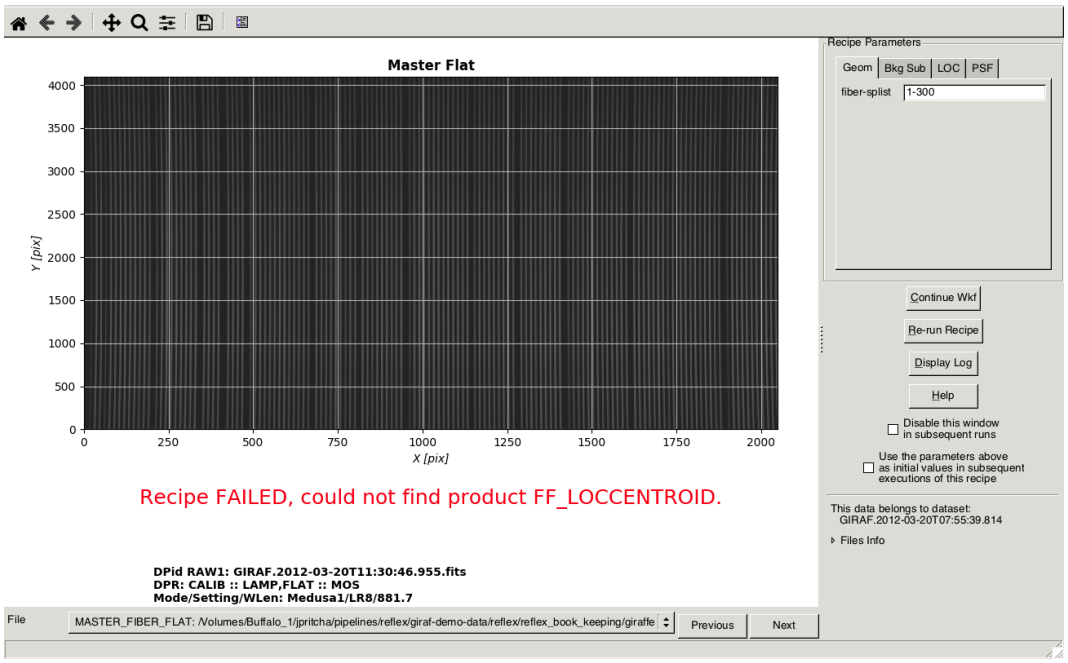


Figure 8.3: *If the MasterFlat recipe fails, click the* `Continue` *button in the rediculously wide error message window to display the products (if any) that the recipe did manage to create before crashing and use the Log file window to try to debug what went wrong, and how to set the recipe parameters in order to successfully reduce the dataset..*

`MASTER_BIAS`). From this the user has then to determine the missing raw data (in this case bias frames). If static calibrations are missing the mechanism unfortunately does not work, but such data should be found by `Reflex` in `<install_directory>/calib/<pipeline_version>/cal`

Table 9.1: A few multifiber spectrographs around the world.

Instrument	Telescope	Observatory	Number of objects	FOV	Resolution
Hectospec	6.7-m MMT	MMT	300	1 deg	1000–2500
6dF	1.2-m UK Schmidt	AAO	150	5.7 deg	
2dF	3.9-m AAT	AAO	400	2 deg	
Hydra	3.5-m WYIN	KPNO	90	1 deg	
FLAMES	8.2-UT2	VLT	135/8	25 arcmin	5600-46000/48000

9 A brief overview of data reduction of multi-fiber spectroscopy data

This section presents a brief description of the reduction of multi-fiber spectroscopic data. If you are a beginner who just got your first data set, this section is probably worth reading. Although the data collected with FLAMES/GIRAFFE is used as an example, the steps outlined here are typical for data reduction of any multi-fiber spectrograph.

Like any other reduction package, the GIRAFFE pipeline has many adjustable parameters allowing to fine-tune the data-reduction. We refer to the pipeline manual to a full description of these parameters.

9.1 Multi-fiber spectroscopy

If you have already had a look at one of your raw science frames, the advantage of using a multi-fiber spectrograph is clear. In one single shot hundreds of objects can be observed. Fibers can be placed at almost any place within the telescope focal plane (within 25arcmin in the case of FLAMES) as shown in Fig. 9.1. This multiplex capability has of course a cost. Due to the limited size of the detectors only a small piece of the spectrum is recorded for each target. Also, the fibers most commonly used in astronomy have poor transmission in the blue region of the spectrum. An advantage in fibres over Multi Object Spectroscopy (MOS) is that there are no issues with slit centering which can impact on radial velocities with slits.

There are number of multi-fiber spectrographs around the world. The main characteristics of some of them are given in Table 9.1.

In the case of FLAMES, the fibers are arranged in a circular pattern around a plate of the size of the telescope focal plane. The fiber end "looking" at the sky has a magnetic button on it. The magnetic side of this button sticks to the plate whereas the other side is open to leave the light of your target to get into the button. In the case of FLAMES the light that enters into the button is deviated into the fibers by a tiny prism. The other end of these fibers are arranged along the long-slit of the spectrograph. Once the light of the fibers get inside the spectrograph, the desired spectral order is selected by order sorting filters. It is then reflected into a double pass collimator and goes to the grating. After an intermediate spectrum is formed, the light is finally re-imaged on the CCD. Although all multi-fiber instruments differ from each other in technical details, the basic idea is the same for all of them.

The basic steps of the reduction of multi-fiber spectroscopy data are the following:

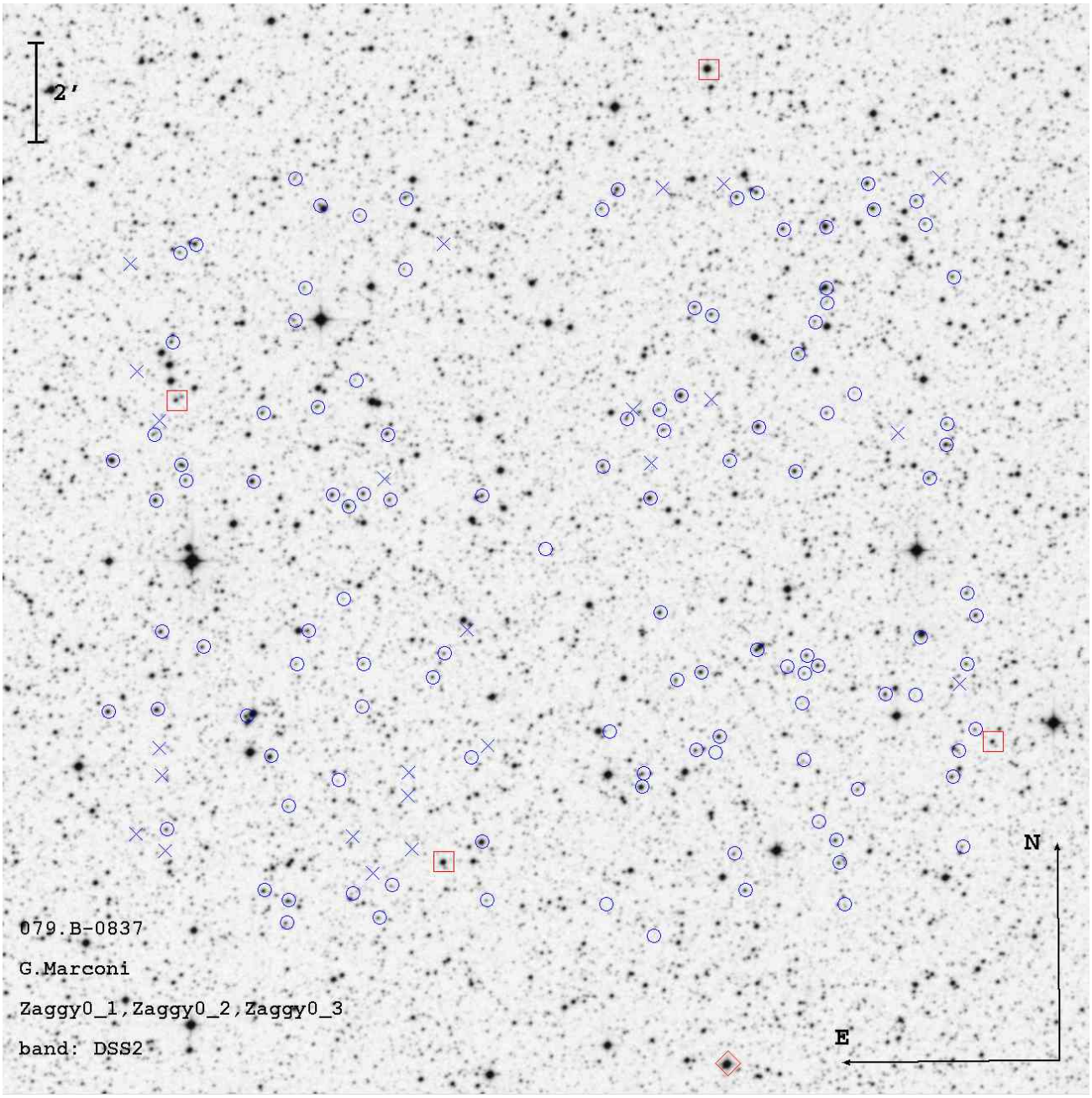


Figure 9.1: An example of the potential of multi-fiber spectrographs. In one shot up to 135 spectra are recorded by GIRAFFE and up 8 by UVES. The figure is a finding chart of a typical FLAMES observation. Circles indicate science targets. Sky positions are marked with crosses and the four FACBs used for centering the field are seen as squares.

- correcting frames for detector cosmetic effects
- determining the location of your data on the detector, i.e., fiber tracing
- extraction of flat-field spectrum and determination fiber transmission
- scattered light correction
- standard star calibration in the case of IFU or ARGUS
- wavelength calibration
- extraction of science data
- sky subtraction

9.2 Correcting detector cosmetic effects

Data reduction of any nature starts by correcting the detector defects referred as cosmetics. These effects and the way to correct them have largely described in different cookbooks[18]. Here we briefly described the main defects.

- **Subtracting the Bias level.** A bias voltage is routinely applied to CCD detectors to ensure that, as near as possible they are operating in a linear manner. This current has the effect that a non-zero count is recorded in all pixels.
- **Subtracting the dark current.** Dark current arises from thermal energy within the silicon lattice comprising the CCD. Electrons are created over time that are independent of the light falling on the detector. These electrons are captured by the CCD's potential wells and counted as signal.
- **Bad pixel correction.** Any detector has a certain number of pixels that are bad, in the sense that these bad pixels record the information inaccurately. This happens because either they are brighter than the others (hot pixels) or because they have low or no sensitivity at all (dead pixels). Bad pixels (or bad columns) are fixed by interpolating the signal in the neighbor pixels (or columns).
- **Cosmic-ray hits.** When a high energy particle hits the CCD, it loses its energy by knocking the atoms constituting the chip itself. That liberates many electrons that cause a bright spot on the image. These high energy particle can either be genuine cosmic rays (exotic particle produced by exploding supernovae, black holes, etc.), or just the product of the decay of some radioactive atoms present in the lenses just above the CCD.
- **Correction of pixel-to-pixel variations.** Pixels in a CCD have all different sensitivities. This means that some of them will convert the light photons more efficiently into electrons than others. Thus an uniform light source like the bright sky or an illuminated screen will not appear uniform on the CCD. This effect is corrected by taking uniformly illuminated images (or flat-fields). Those images are used to construct a sensitivity map of the CCD.

[18] A good starting point is the cookbook *A User's Guide to CCD Reductions with IRAF*, by Philip Massey which can be found in the IRAF website http://iraf.noao.edu

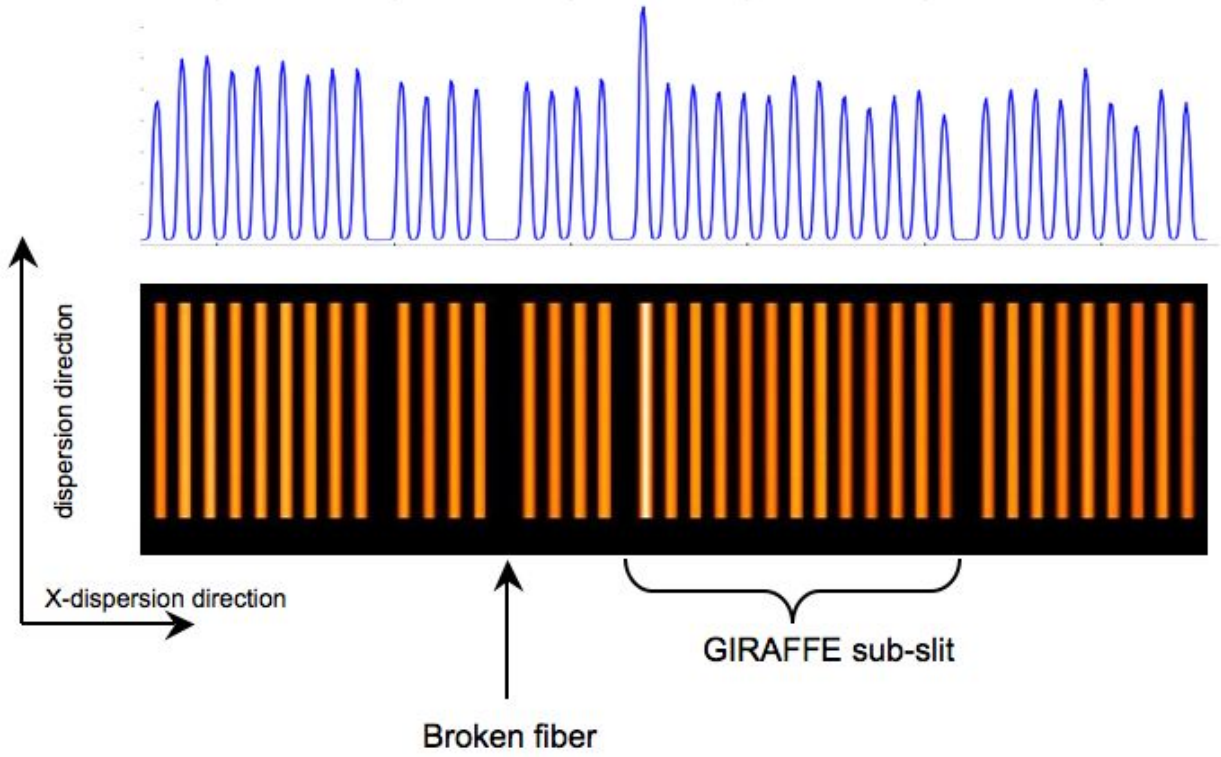


Figure 9.2: An extract of a raw image of a flat-filed frame is shown the bottom panel. The sub-slits (packets of fibers) defined in the previous figure are clearly seen in the image as well as a broken fiber. In the top panel, a cross section of the frame is shown where the nearly Gaussian profile of the fibers can be seen.

In the case of spectroscopic data, the first three steps are carried out in the same way as done in imaging data reduction whereas cosmic ray cleaning and flat-field corrections are not. The correction of these two effects will be discussed in the next sessions.

9.3 Fiber localization and tracing

As described above and in Fig. 9.2, the fibers are arranged side by side along the spectrograph slit. After being dispersed by the grating, the spectrum of each is recorded on the CCD also side by side. The direction along which the light is dispersed is called the dispersion direction. The direction perpendicular to the dispersion is called the cross-dispersion direction (or spatial direction in slit spectroscopy). These directions are also indicated in Fig. 9.2.

Thus the first task in the data reduction process (after cleaning the detector defects) is to know where

the spectrum of each fiber actually is on your 2 dimensional CCD. This processes is called fiber localization.

First, a exposure with all fibers uniformly illuminated by a calibration lamp is taken. This same exposure will be used to flat-field the data later. Then a line is cut along the cross-dispersion direction. In the top panel of Fig. 9.2 we see a series peaks more or less evenly spaced. Each of these peaks corresponds to a fiber. In many pipelines, the fiber profiles is approximated by a Gaussian function. The pipeline fit each of those peaks with a Gaussian function and stores for each fiber its center and width.

In both panels of Fig. 9.2 we can easily distinguish three packets of fibers with a larger gap in between. Each packet represents a GIRAFFE sub-slit. We might also find a gap within a given packet. This happens when a fiber is broken. In the bottom of Fig. 9.2 we show an extract of a raw image of a flat-field exposure. Three packets of fibers are seen. In the second one, there is a missing (broken) fiber.

In order to deal with broken fibers, the pipeline uses the fact that the size of the fibers is known and the instrument is stable to a point that the center of the fibers don't move by more than 1 pixel. So the pipeline knows where a given fiber should lie and if the localization algorithm cannot measure any signal there that fiber is declared as broken.

The second step once the initial position of the fibers is known is to determine the fiber profile along the dispersion direction. Using the initial position for a given fiber, the pipeline moves a couple of pixels along the dispersion direction and, again, it carries out a Gaussian fit at this new position. A new center and width are found. This is repeated until the edge of the CCD is reached. At the end the pipeline determine a sort of tube or tunnel where the science data will be recorded. An example of these tubes are shown at Fig. 9.3.

9.4 Extraction, flat-field spectra and fiber transmission

Once these "tubes" have been determined we can extract the signal on the CCD. The first thing to be extracted using the same flat-field frame is the flat-field spectrum.

From Fig. 9.2 we know already that the signal spreads over many pixels. In the case of GIRAFFE, the MEDUSA fiber profile is spread over 6 pixels. There are two ways of summing the information spread over the fiber profile. In the simplest case we add up all pixels inside the fiber profile. This is what is called standard or summed extraction.

The standard extraction ignores the fact that there pixels which contains more counts (better quality information) than others. They all contribute with equal weight to the final spectrum. Since the noise associated to each pixel is given by the squared-root of the number of counts on this pixel (Poisson noise), we can easily see that give the same weight to pixel with lower counts means that we are adding noise to our final spectrum.

The shape of the fiber profile can be used as a weight function, thus instead of a simple addition we weight its flux by its noise. In this way, better pixels will give a higher contribution to the final spectrum. This is called optimum extraction (e.g., Horne, 1986, PASP 98, 609). Note that at present optimal extraction is only present for Medusa and not for Argus or IFU.

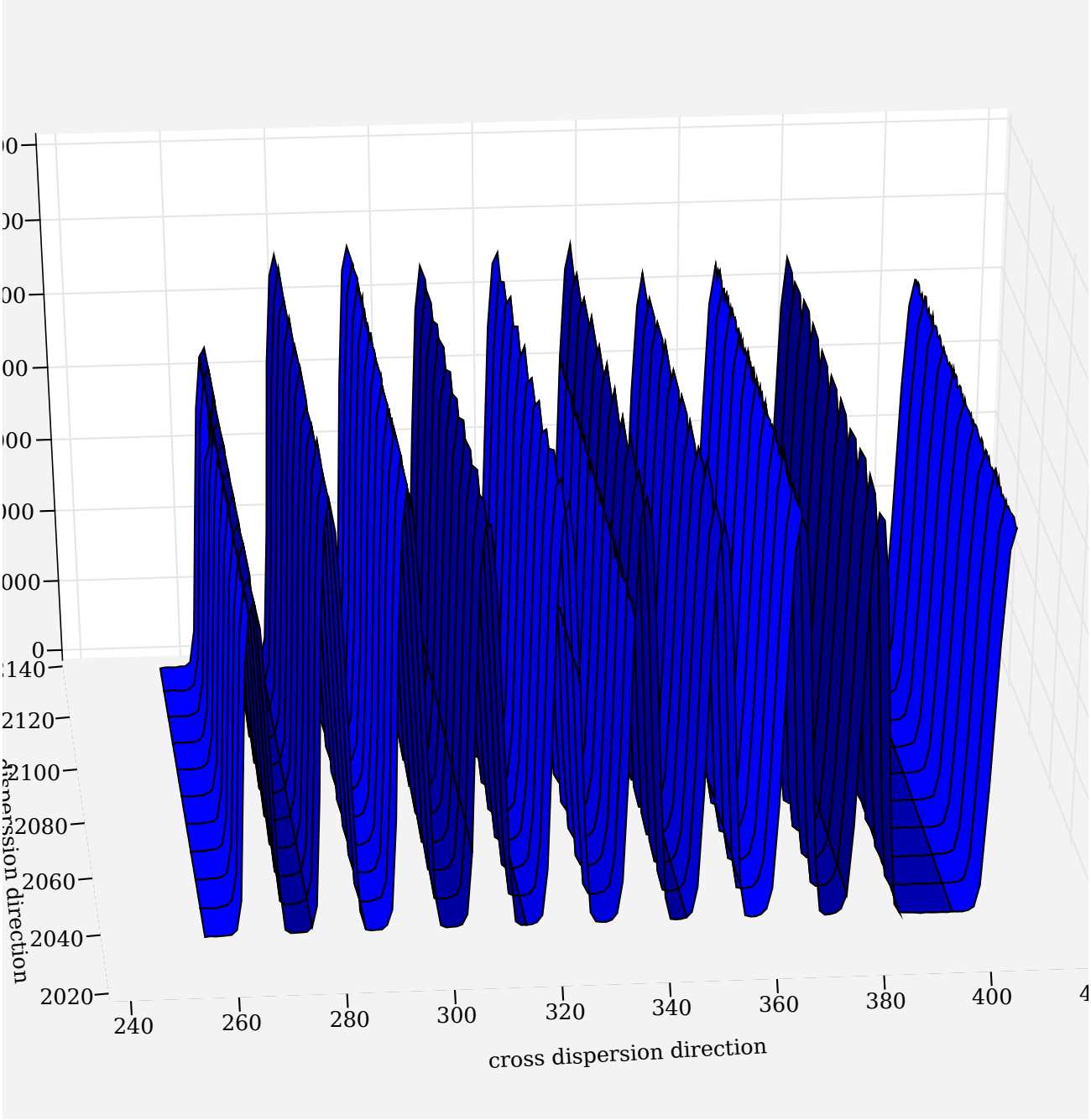


Figure 9.3: 3D representation of the first packet seen in Fig. 9.2 showing the Gaussian tubes.

Table 9.2: GIRAFFE fiber transmission. The values given all losses, focal ratio degradation, optics and coupling. For wavelengths redder than 600nm the transmission is constant. Variations of a few percent between different fibers are measured (see Pasquini et al. 2003).

Fiber type	370nm	400 nm	450 nm	600 nm
MEDUSA	0.47	0.52	0.55	0.61
ARGUS	0.52	0.58	0.62	0.70
IFU	0.49	0.55	0.58	0.66

The optimum extraction has an additional advantage with respect to the standard extraction. Since we know that the distribution of the intensity of the pixels should follow a smooth and continuous function, any pixel deviating a few per cent of this profile is likely to be cosmic-ray! The pixel hit by a cosmic ray can be replaced by the interpolation of its neighbors cleaning the final spectrum.

Extraction of the spectrum of the flat lamp has two main functions. The first is to correct the pixel-to-pixel variation in our science data. Second, the amount of light entering the fibers is supposed to be similar. Thus any difference of the intensity of the extract flat field spectrum is due to differences in the fiber transmission.

In imaging or even in slit spectroscopy, one can carry out a two-dimensional flat-field correction. This means that (after some manipulation) your science frame can be divided by the flat-field image.

In the case of fiber spectroscopy we have seen that the intensity of the pixels drops quickly at the edge of the fiber profile. If the two frames are slightly miss-aligned, (i.e., the two profiles don't match exactly each other), the division will produce an sort of parabola instead of a flat-image.

Flat-field are corrections are done in one dimension, i.e., the extracted science data is divided by the flat-field spectrum. In this way we avoid introducing artifact due to the mismatch of the science and the flat-field.

Fibers are not perfect devices. A certain amount of photons that enter in one end don't make it to the other end of the fibers. The amount of lost photons depends of their energy (or wavelength). Typical transmissions as a function of wavelength for the different fiber systems of FLAMES/GIRAFFE are given in Table 9.2. Values are taken form Pasquini et al. (2003, SPIE 4841, 1682)[19].

Now if you consider a set of fibers sharing the same characteristics (like the MEDUSA fibers in FLAMES, for instance), although they have a similar behavior, they are not exactly similar to each other. Some of them carry light better than others.

In a flat-field frame, the amount of light entering the fibers is assumed to be the same for all fibers. Thus comparing the intensity of the extracted flat-field spectra, we can derive what is called the fiber relative transmission.

This is important when one wants to do additive operations with the fibers and critical in operations like sky subtraction as described in Sec. 2.8 and in Wyse & Gilmore (1992, MNRAS 257, 1).

[19]This paper is available at http://www.eso.org/instruments/flames/doc/spie.ps

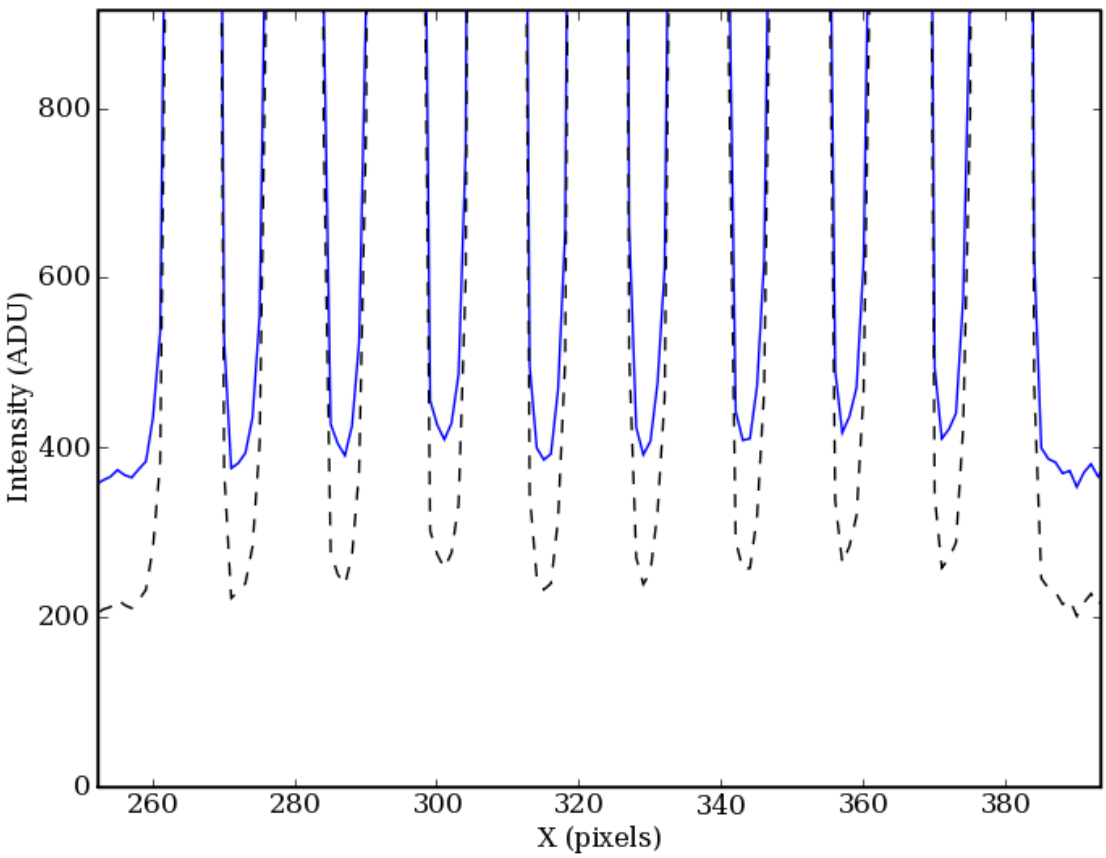


Figure 9.4: Cut across the fibers. Solid and dashed lines show the minimum level before and after bias subtraction. The remaining ADUs seen in the case of the bias subtracted frame are due to the dark current and scattered light.

9.5 Scattered light correction

A better idea of what scattered light is given in Fig. 9.4. In this figure we show a zoom-in of the base of a packet of fibers. The solid line and dashed lines represent the fiber profiles before and after the bias subtraction. We see that even after the bias removal the signal doesn't go to zero. This remaining signal is the scattered light.

This is because part of the light is scattered inside the spectrograph. This scattered light has two components. A smooth one, covering the whole CCD which is proportional to the amount of light entering the spectrograph.

A second component is a local one and it is caused by the presence of bright objects (or a simultaneous comparison lamp). In this case, it might happen that the charges of the CCD will jump to the neighbor pixels.

The smooth component is easy to subtract. A two dimensional fit is carried out on the whole CCD using the points of the detector in the gap between two adjacent fibers.

The local component might require much detailed look in the light in the inter-fiber regions to determine whether or not this is an issue. The local component of the scattered light behaves like an extra continuum (i.e., with no spectral features) whose spectral energy distribution follows the one of the object causing the scattered light.

A good correction of the scattered light is essential to achieve an accurate sky subtraction.

9.6 Wavelength calibration

If you look at your spectrum after extraction you might already recognize a few features on it (Hydrogen lines, Li in the case of young stars, etc.). But having that in pixel space is pretty much useless. This is what the wavelength calibration lamp does. Wavelength calibration is achieved using a Hallow-Cathode-Lamp.

An HCL usually consists of a glass tube containing a cathode made of the material of interest, an anode, and a buffer gas (usually a noble gas). A large voltage across the anode and cathode will cause the buffer gas to ionize, creating a plasma. These ions will then be accelerated into the cathode, sputtering off atoms from the cathode. These atoms will in turn be excited by collisions with other atoms/particles in the plasma. As these excited atoms decay to lower states, they will emit photons, which can then be detected and a spectrum can be determined.

The wavelengths of the emission line spectra of these lamps are known from laboratory tests. From our ThAr frame, we measure the (x,y) position on the CCD for the emission lines. From an atlas of emission lines[20] we can associate a pixel to a wavelength. By means of a polynomial fit we can compute the transformation function from pixel to wavelength space, $\lambda \rightarrow f(x,y)$.

9.7 Extraction of the science

The science data is extracted in the same fashion as described above for the flat-field. After extraction, the scattered light is removed, the science spectrum on each fiber is divided by its respective flat-field spectrum, correct for the fiber transmission variants and the keywords containing the information about the wavelength calibration are added to the fits header of the image.

Since the description of these keywords vary from package to package, in most of the cases, a process called rebin is carried out in which we resample our spectra in order to have a constant step in wavelength ($\Delta\lambda = cte$). The keywords used describing an evenly sampled spectrum obey the FITS standards and therefore is the same regardless the data reduction package you are using. Also, rebinned spectra can be easily read as a vector by your own programs written in FORTRAN, C, python, etc.

Your spectra are ready to be analyzed.

[20] NOAO provides Spectral Atlases for different lamps at http://www.noao.edu/kpno/specatlas/index.html

9.8 Sky subtraction

In the case you are dealing with faint source whose signal is close to the read-out noise of the CCD, or with data from the redder wavelength settings, you might want/will probably need to carry out sky subtraction. With some care, sky subtraction as good as 1–3% can achieved. This requires:

- proper bias and dark correction
- scattered light correction
- fiber-to-fiber transmission

Before submitting a proposal PIs should consider downloading previous GIRAFFE spectra from the ESO archive to see how badly their spectra will be contaminated by sky lines. Figures 9.5 and 9.6 show how strong sky lines can be in the red.

We note that the current version of the GIRAFFE *pipeline* does **not** perform sky subtraction, however sky-subtraction has been implemented in this Reflex *workflow* using python scripts outside the actual CPL pipeline itself. This implementation is fairly simple[21] but nonetheless there are a number of parameters to consider and one should experiment with the various options for the sky treatement in order to find the setting that best suit your data. A more detailed extended demonstration of the sky subtraction is given in section 10.3.

PIs should consider referring to the following articles (amongst others) on how to remove sky features in FLAMES data:

- Battaglia, *et al.* (2008)[1]: Contains a detailed description of how sky lines can removed from FLAMES-GIRAFFE spectra.
- Koch, *et al.* (2007)[7]: An estimate in the final accuracy of sky subtraction of ∼3 per cent is given for Leo spectra.
- Koch, *et al.* (2006)[6]: An estimate in the final accuracy of sky subtraction of ∼2 per cent is given for Carina spectra.

Finally, we refer to Wyse & Gilmore (1992) [10] for a very good discussion in the problematic of achieving accurate sky subtraction and how to assess the quality of the scattered light correction and the final sky subtraction using the inter-fiber regions and the broken fibers.

[21] It does not, for example, allow for possible wavelength offsets between the individual science and sky spectra, or allow for a global scaling of the sky spectra to each science spectrum which may improve the quality of the sky subtraction.

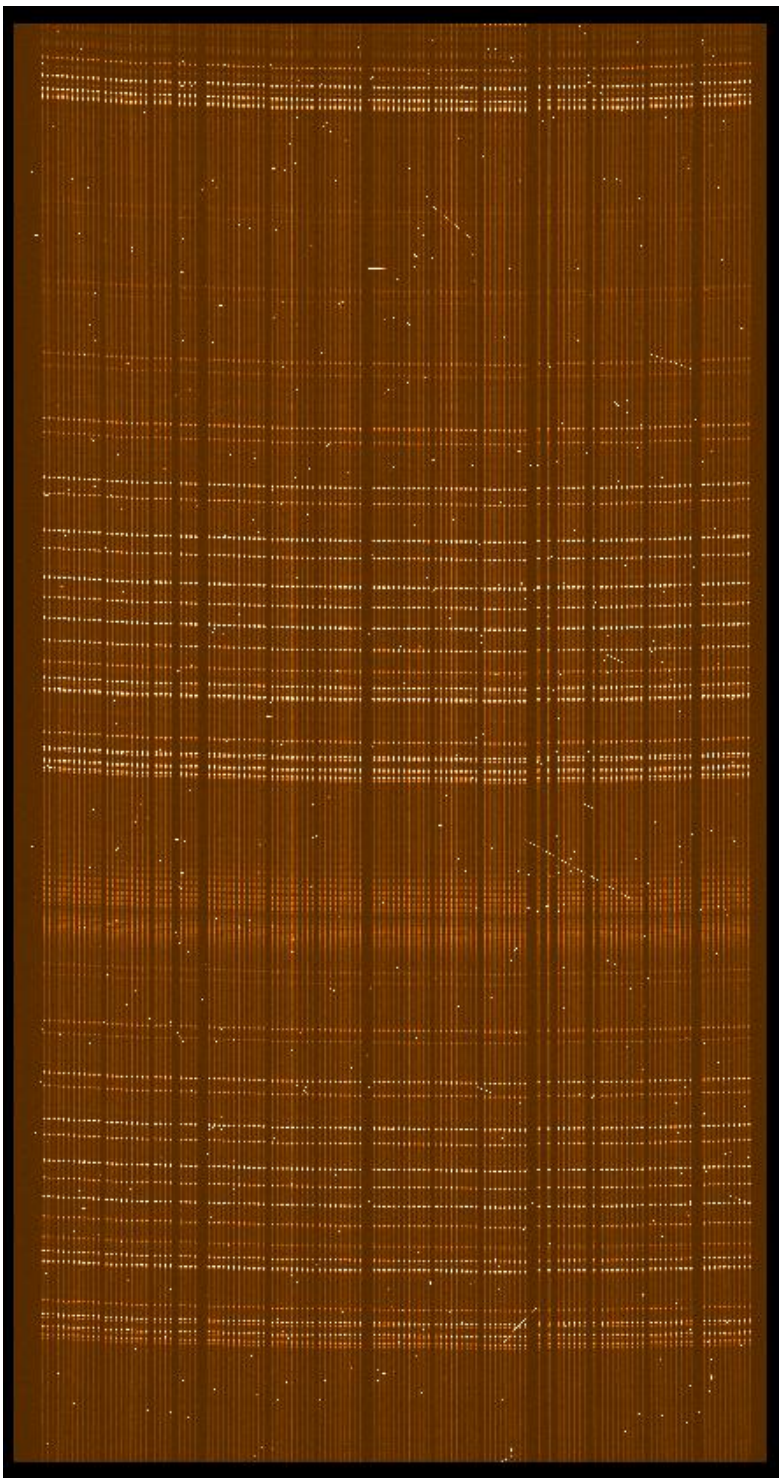


Figure 9.5: This image shows how, especially in the red, there are many sky lines. Removing them can be critical to obtaining good science data. The exposure was taken using GIRAFFE at L881.7-nm for 2750-s.

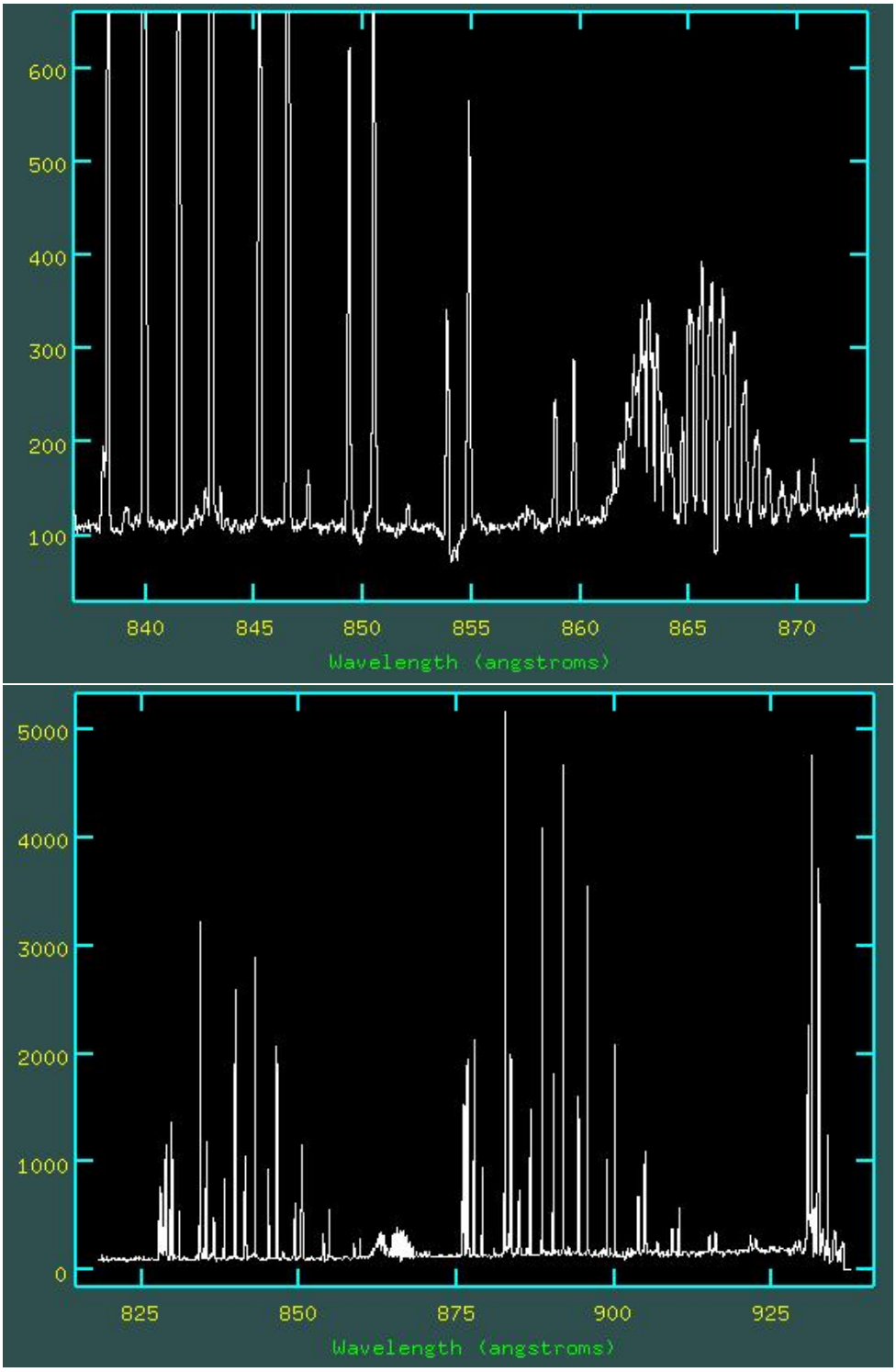


Figure 9.6: Extracted spectra of the image in Fig. 9.5 showing a number of bright sky emission lines.

10 An extended Demo

This section presents an extended DEMO showing how to enable and use Cosmic Ray cleaning, how to use the Sky Subtraction features and then some problems in the data that may arise and how to deal with them.

10.1 The Science and Standard GUIs in detail

Compared the previous versions of the GIRAFFE workflow, the science and standard star GUIs now have quite a number of widgets in the lower half of the window which are used to control the sky subtraction and the spectral display. Figure 10.1 shows an annotated view identifying the various widgets and groups of widgets that will be referred to and explained below.

By default, when the science/standard GUIs open, they display the 'Signal' and 'Signal-Sky' spectra of the fibre/object with the highest measured S/N ratio in the Spectrum Display panel.

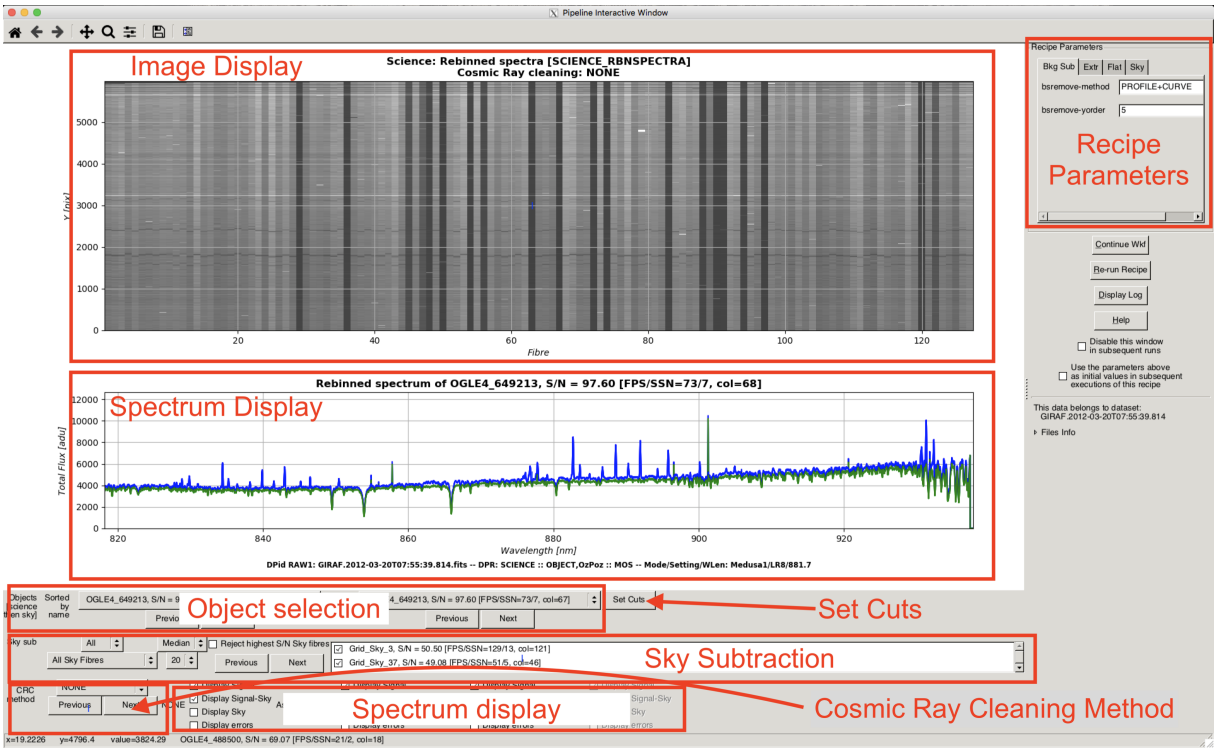


Figure 10.1: *The Science Interactive GUI with the various features related to sky subtraction annotated.*

10.2 Cosmic Ray Cleaning

Cosmic Ray Cleaning [CRC] has been implemented via two external Python packages:

- PyCosmic: https://sourceforge.net/projects/pycosmic, see also http://www.bhusemann-astro.org/?q=pycosmic.
- Astro-SCRAPPY: https://github.com/astropy/astroscrappy.

Both packages are based on Pieter van Dokkum's L.A.Cosmic algorithm, [9].

When the **Grouped** parameter (in the 'Data reduction strategy parameters' section of the main workflow is set to `tpl` or to `night`, a third cosmic ray correction algorithm is applied, namely medianing; each set of N input science files grouped together will be medianed into a single file which will then be processed as a single file by the `giscience` recipe. In such a case, for the other CRC methods (Astro-SCRAPPY and PyCosmic) the N individual files will be cosmic ray corrected and then submitted to the `giscience` recipe which will first take a pixelwise mean of the N frames which it will then process as a single frame.

Limited evaluation of the two python package based CRC methods (Astro-SCRAPPY and PyCosmic) suggests that the two packages seem to produce similar, but not exactly equal results. PyCosmic seems to detect and correct more cosmics than Astro-SCRAPPY, presumably because it employs a multi-pass approach and has been improved for fibre-fed spectrograph CCD-image, but at the cost of significantly longer execution times.

We leave both methods enabled by default so that you can decide which is best for your purpose.

You can disable one or other method either by removing the software from your system, or at least 'hiding' it from Reflex, or by setting one or other (or both) of the `EnablePyCosmic` or `EnableAstroScrappy` variables to false inside the 'Cosmic Ray Cleaning' composite actor.

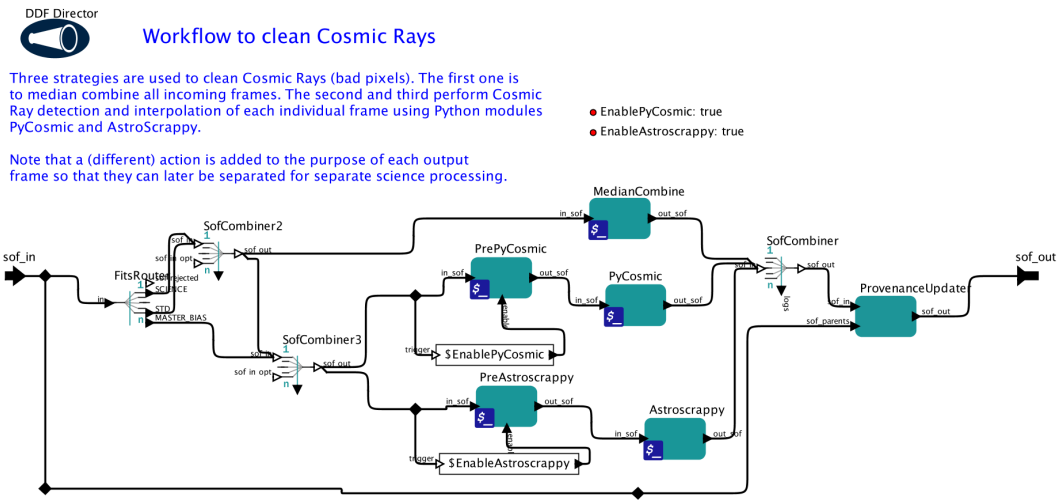


Figure 10.2: *The Cosmic Ray Cleaning composite actor. The* `EnablePyCosmic` *or* `EnableAstroScrappy` *variables are in the top right corner, just double click on them to open a dialog window in which you can set their values..*

10.2.1 PyCosmic

From http://www.bhusemann-astro.org/?q=pycosmic:

> *PyCosmic is a tool to detect and clean single images from the disturbing cosmic ray hits. It uses Laplacian-edge detection scheme of L.A.Cosmic, but extends the algorithm specifically for fiber-fed spectrograph CCD images. The software was designed and tested specifically for the data of the Calar Alto Large Integral Field Area (CALIFA) survey that is using PMAS fibre-fed integral-field spectrograph. The program was also tested on other data including classical long-slit observations. The current version is written in Python and can be executed from the command-line or included in Python scripts.*

If you use PyCosmic to reduce your data, please cite [5].

10.2.2 Installing PyCosmic

If PyCosmic is not (yet) available for your system via a package manager[22] then you will need to install the package manually (see below). If it is available, just install it from the package manager and then it should be available to the GIRAFFE Reflex workflow.

If not available via a package manager we recommend installing via `pip`.[23] Furthermore to avoid possible eventual conflicts with system files, we recommend a *user* installation.

```
pip install --user \
  https://downloads.sourceforge.net/project/pycosmic/PyCosmic_v0.5.tar.gz
```

10.2.3 Astro-SCRAPPY

From https://github.com/astropy/astroscrappy:

> *Astro-SCRAPPY is designed to detect cosmic rays in images (numpy arrays), based on Pieter van Dokkum's L.A.Cosmic algorithm.*
>
> *Much of this was originally adapted from cosmics.py written by Malte Tewes. I have ported all of the slow functions to Cython/C, and optimized where I can. This is designed to be as fast as possible so some of the readability has been sacrificed, specifically in the C code.*

If you use Astro-SCRAPPY to reduce your data please cite the Zendo DOI: https://zenodo.org/record/1482019 and the original van Dokkum paper [9].

[22] MacPorts for macOS, RPM for Linux

[23] On systems with more than one version of python installed/available, there is (usually) also a corresponding version of pip installed/available for each version of python. As esoreflex currently uses Python version 2.7, please insure you are using the 2.7 version of `pip` which is possibly `pip2`, `pip2.7`, `pip-2.7` or similar.

10.2.4 Installing Astro-SCRAPPY

If Astro-SCRAPPY is not (yet) available for your system via a package manager [24] then you will need to install the package manually (see below). If it is available, just install it from the package manager and then it should be available to the GIRAFFE Reflex workflow.

If not available via a package manager we recommend installing via `pip`.[25] Furthermore to avoid possible eventual conflicts with system files, we recommend a *user* installation.

```
pip install --user astroscrappy
```

10.2.5 Enabling Cosmic Ray Cleaning

If you could install the Python Cosmic Ray cleaning packages with the same package manager as you installed Reflex with (MacPorts for macOS/RPM for Fedora) or via *pip* as described above then CRC should now work for you 'out-of-the-box', and you can just re-run the workflow (e.g. on the demodata) and you should get Cosmic Ray cleaned results (see section 10.2.6).

10.2.6 Cosmic Ray Cleaning results

Assuming both CRC Python packages are installed and if necessary Reflex has been configured, then if you re-run the GIRAFFE workflow on the demodata then for each science or standard file the science/standard recipe should run three times, once for the non Cosmic Ray Cleaned original RAW file, once for the Astro-SCRAPPY cleaned raw file and once for the PyCosmic cleaned raw file.

Apart from the CRC and science/standard actors taking somewhat longer to execute than previously (and most of the others executing rather quickly if you already ran the workflow on the demodata because they are running in lazy mode) there should be no obvious difference in the way the workflow proceeds, except when you get to the Science or STD star Interactive GUI windows, but even there, at first sight, nothing will seem different. This is because, by default the non CRC 2D image is displayed in the image display when the GUIs first open.

However now, in the 'Cosmic Ray Cleaning Method' selector, at the bottom left of the window (see figure 10.1) you should find three possible methods to select from, the NONE (non) CRC version, the Astro-SCRAPPY cleaned version and the PyCosmic cleaned version. As you select the different versions (via the drop-down menu or the 'Next' and 'Previous' buttons) the different versions of the 2D-extracted image will be displayed in the Image Display panel (see figure 10.3).

A more objective indication of the merits of CRC can be seen in the extracted spectra, i.e. in the Spectrum Display panel. Figure 10.4 shows the No CRC, Astro-SCRAPPY and PyCosmic versions

[24] MacPorts for macOS, RPM for Linux. At the time of writing, Astro-SCRAPPY is available as a Fedora RPM on Fedora 25 and newer.

[25] On systems with more than one version of python installed/available, there is (usually) also a corresponding version of pip installed/available for each version of python. As esoreflex currently uses Python version 2.7, please insure you are using the 2.7 version of `pip` which is possibly `pip2`, `pip2.7`, `pip-2.7` or similar.

of the extracted spectra of the default selected object[26] from the *SN 1604* dataset.

[26] By default, when the science/standard GUIs open, they display the fibre/object with the highest measured S/N ratio.

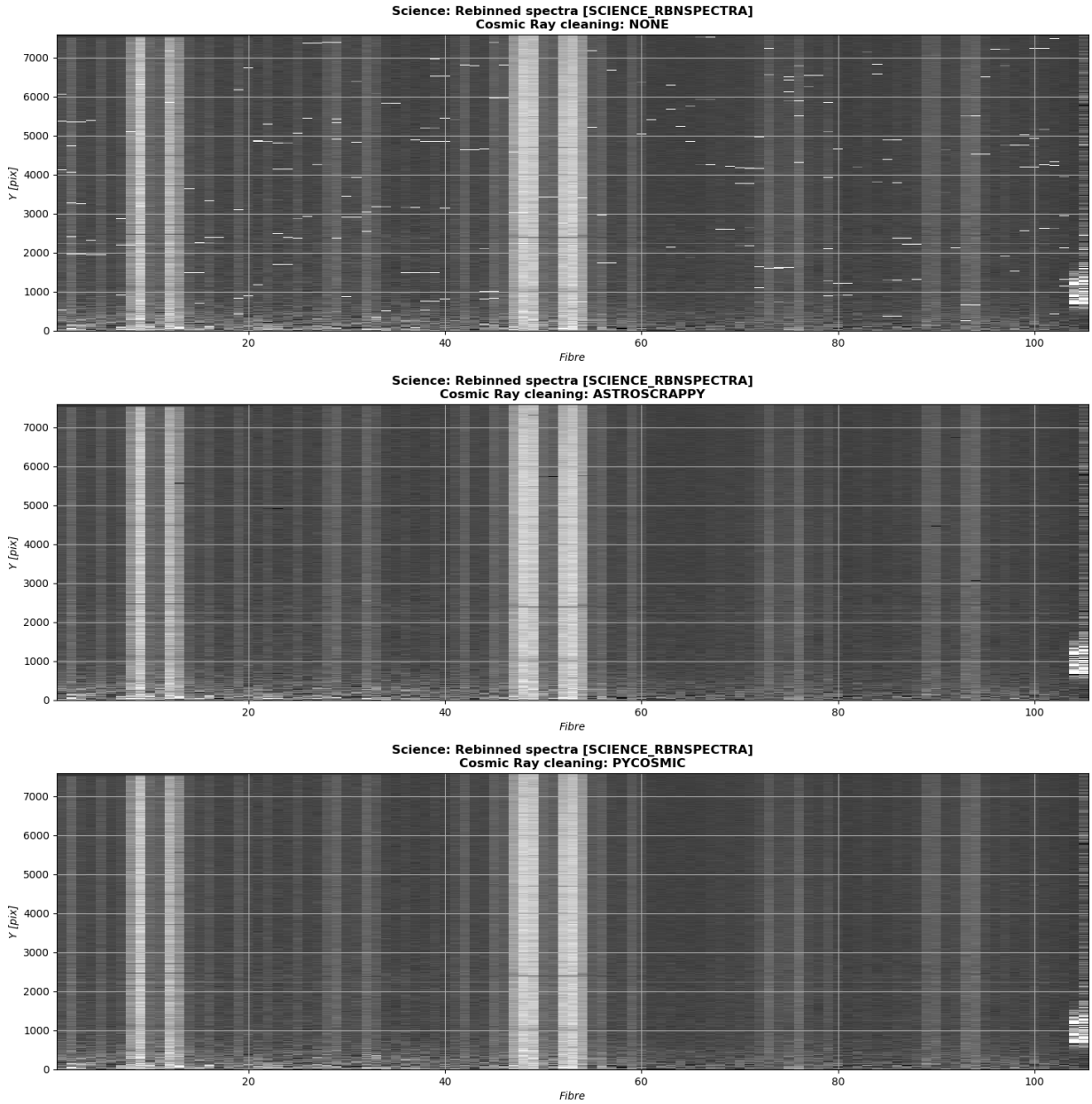


Figure 10.3: *The Image Display panels from the Science Interactive GUI for the 'SN 1604' dataset displaying the NONE (non) Cosmic Ray Cleaned product (top), the Astro-SCRAPPY cleaned version (middle) and the PyCosmic cleaned version (bottom). To the eye, both Cosmic Ray Cleaning implementations provide an obvious improvement.*

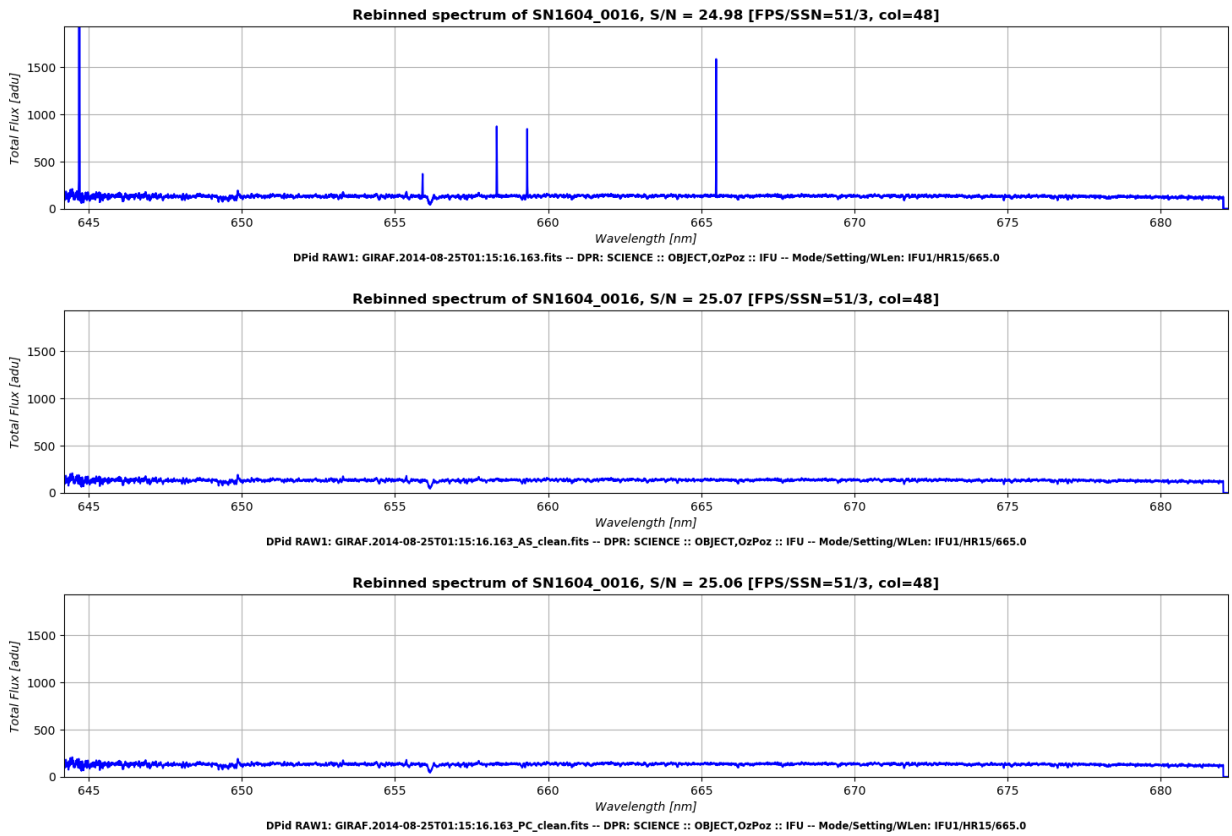


Figure 10.4: *The Spectrum Display panels from the Science Interactive GUI for the 'SN 1604' dataset displaying the NONE (non) Cosmic Ray Cleaned product (top), the Astro-SCRAPPY cleaned version (middle) and the PyCosmic cleaned version (bottom). These plots are obtained by deselecting all but the 'Signal' plot of the respective CRC method in the Spectrum Display sub-panel of the GUI window (see figure 10.1).*

10.3 Sky Subtraction

Basic sky-subtraction has been implemented in this Reflex *workflow* using python scripts outside the actual CPL *pipeline* itself, hereafter referred to as the sky-subtraction module. This means that this functionality is ONLY available in the workflow. In particular it is NOT available when running the pipeline from the command line directly with esorex (nor in Gasgano).

This sky-subtraction module implementation is fairly simple and has not yet been subjected to thorough and rigourous scientific validation. It is thus offered as a 'potentially useful tool' to allow you to 'explore the possibilities', but without any warranty of being 'fit-for-purpose'.

Having said that, even the default settings will probably be better than nothing, at least for faint sources and/or the redder wavelength settings.

10.3.1 Known Limitations

This section describes the *known* limitations, there are almost certainly many other unknown limitations which we haven't even thought about yet.[27]

1. It does not allow for possible wavelength offsets between the individual science and sky spectra,
2. It does not allow for a global scaling of the sky spectra to each science spectrum which may improve the quality of the sky subtraction.

If any of these limitations are important in your data, you can of course deal with them outside Reflex, and then in that case, the sky spectra produced by the workflow may at least be a good starting point for dealing with such issues.

10.3.2 Method: basic description

GIRAFFE fibres assigned to sky are processed by the GIRAFFE pipeline in the same way, and at the same time, as fibres assigned to science targets (and possibly simultaneous wavelength calibration fibres) and are included in the pipeline product files along with the science (and possibly simultaneous wavelength calibration) fibres.

The sky-subtraction module first identifies the fibres, if any, assigned to sky. It then calculates a sky-spectrum for each science fibre based on all, or a subset, of the available sky-fibres. The exact method used to calculate the sky spectrum is determined by the values of seven parameters, which you can find in the 'Sky' tab of the 'Recipe Parameters' subpanel of the science/standard GUIs.[28] These parameters can also be controled more 'intuitively' in the 'Sky subtraction' sub-panel of the GUI window (see figure 10.1).

[27] You are actively encouraged to email *https://support.eso.org* to point out any and all actual or potential, as-yet-undocumented, limitations you might discover or think of.

[28] Even though they are not technically parameters of the giscence CPL recipe, but parameters of the sky-subtraction module, at the GUI level they are handled in a similar way by the workflow.

As any of these parameters are changed, the sky-spectrum for each science fibre is re-calculated 'on-the-fly' and when finished the spectrum display is updated, so that you can se the effect of adjusting the parameters in 'realtime'.

Like the true recipe parameters, if you re-run the recipe (because you have changed one or more of the true recipe parameters) the sky-spectra in the next iteration will be calculated with the current values of the sky-subtraction parameters. But if you want the current sky-subtraction parameters used for the next dataset(s) you need to click the 'Use the parameters above...' radio button, before clicking the 'Continue Workflow' button.

The seven parameters, and their meanings are:

1. **sky-applyTo**: Select between `All|None`. Selecting 'None' effectively turns off sky-subtraction. 'All' means that that the same method (as determined by the following six parameters will be applied to all of the fibres.

2. **sky-method**: Select between `All Sky Fibres|Best <N> Fibres by S/N|Nearest <N> Fibres`:

   - `All Sky Fibres`: The additional filtering algorithms below[29] are applied to all sky-fibres, the **sky-numFibs** algorithm has no effect on the selection for this option. The result is one and only one sky-fibre selection list which is used for calculating the sky-spectrum for all science spectra, i.e. there will be one common sky-spectrum applied to all science spectra.
   - `Best <N> Fibres by S/N`: The additional filtering algorithms below are applied to the <N>=**sky-numFibs** sky-fibres with the highest measured S/N. Again the result is one and only one sky-fibre selection list, though different from the one resulting from `All Sky Fibres` if **sky-numFibs** is less than the total number of allocated sky fibres. The same list is used for calculating the sky-spectrum for all science spectra, i.e. there will be one common sky-spectrum applied to all science spectra.
   - `Nearest <N> Fibres`: The additional filtering algorithms below are applied to the <N>=**sky-numFibs** sky-fibres closest on the sky to each science fibre. If **sky-numFibs** is less than the total number of allocated sky fibres, there will be a sky-fibre selection list generated for each individual science fibre. Thus in general, but depending on the number of science fibres, the number of sky fibres and their respective positions on the sky, there could be a different sky-fibre selection list, and thus a different calculated sky-spectrum for each and every science fibre.

3. **sky-statistic**: Select between `Median|Mean`: The sky value at each wavelength bin is calculated as the median or mean of the values of the corresponding wavelength bin of fibres in the sky-fibre selection list. In both cases the error at each wavelength bin is calculated as the *mean* of the errors of the corresponding wavelength bin of fibres in the sky-fibre selection list.

4. **sky-numFibs**: Select the number of fibres to include in the sky-fibre selection list.

5. **sky-reject**: This option toggles on and off the application of the non-inclusion (i.e. rejection) of the few sky fibres with the highest signal levels, as recommended by [10]. We use the ,easure

[29] **sky-reject** and **sky-obj-ignore-list**.

signal to noise level, rather than the direct signal level. The number of fibres is set by the **sky-reject-thresh** parameter, depending on the number of sky fibres available.

6. **sky-reject-thresh**: A list of thresholds at which to reject an additional fibre, if **sky-reject** is True/toggled-on. It is perhaps easiest to explain by way of example. The default value for this parameter is 8,12. This means that if there are between zero and seven sky-fibres available, none will be rejected. If there are between eight and eleven sky-fibres available, then one sky-fibre with the highest S/N will be excluded from the sky-fibre selection list. And if there are twelve or more sky fibres available the two fibres with the highest S/N will be excluded. If one would add a third number to the list, e.g. 8,12,15, then a third fibre would be excluded if and when fifteen or more sky fibres are available.

7. **sky-obj-ignore-list**: Specific sky fibres can be excluded from the sky-fibre selection list by name. The names can be specified as a comma separated list in the Recipe Parameters Sky tab sky-obj-ignore-list field, or by un-checking the corresponding entry in the list box in the 'Sky subtraction' sub-panel of the GUI window. This feature is particularly useful when a specific sky fibre allocation position is not a valid sky spectrum (e.g. because it is contaminated by a star), especially when the same fibre is used in multiple datasets.

When you click the 'Continue Workflow' button, the sky-spectra and the sky-spectra-errors are saved to the RBNSPECTRA and RBNERRORS product FITS files in appended extensions named 'SKY' and 'SKYERRS' respectively.[30]

If you disable the the Science GUI display, at anytime, the sky calculation will still be made and the results will still be written to the product files.

10.3.3 Method: error propagation

The pipeline calculates the error of each wavelength bin of the extracted spectra, science, sky and simultaneous wavelength calibration fibres alike.

As noted above, the error of each wavelength bin of the sky spectra is is calculated as the *mean* of the errors of the corresponding wavelength bin of individual fibres in the sky-fibre selection list.

The error of each wavelength bin of the Signal-Sky spectra is is calculated as the the simple *sum* of the Signal and Sky errors.

10.3.4 Example

Figure 10.5 shows a comparison of the 'Signal' and 'Signal-Sky' spectra for the first dataset.

[30] If you have enabled the creation of SDP-style products, the SKY and SKYERRS are written to additional columns of the corresponding SDP-style FITS tables created for each science target fibre.

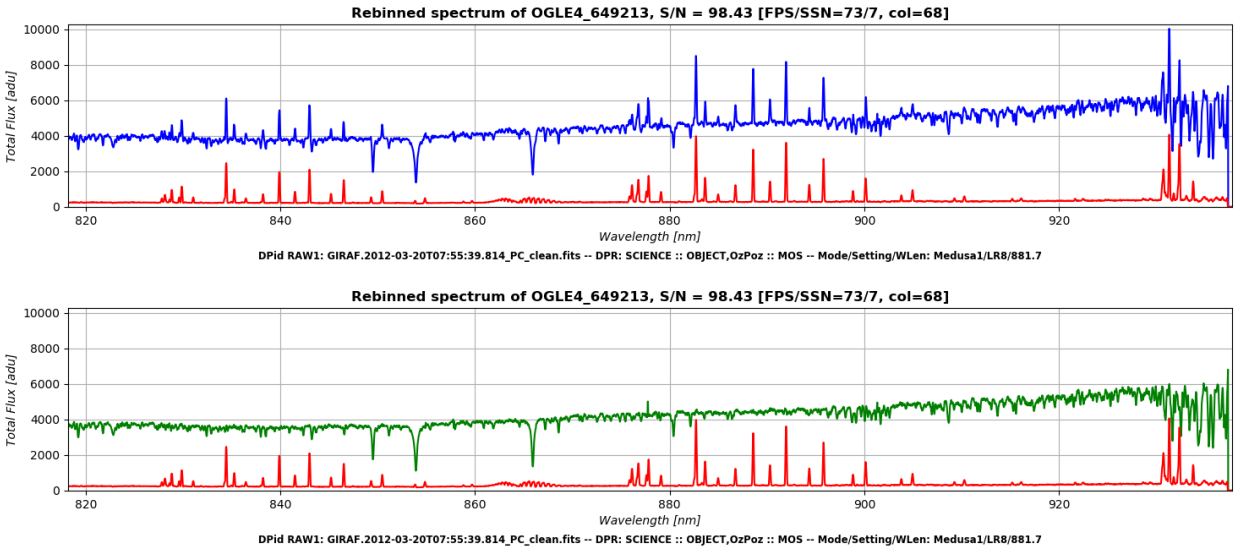


Figure 10.5: *The Spectrum Display panels from the Science Interactive GUI for the first dataset displaying the 'Signal' (top) and 'Signal-Sky' (bottom) (i.e. sky-subtracted) spectra for the PyCosmic cleaned processing.*

10.4 Creating a new *Slit Geometry Table*

In the past at least, it was sometimes necessary to re-create the Slit Geometry table. However, during the preparation of this tutorial document it was not possible to find an input Wavelength calibration file that presented the problem, even the one that was used to illustrate the problem in the old cookbook was processed successfully with the default parameters of the pipeline and the Master Slit Geometry Table included in the static calibrations for the current pipeline distribution.

So if you do come across a dataset with a wavelength calibation which presents a result like that illustrated in figure 10.6, please contact *https://support.eso.org* so that the appropriate method to solve this issue with the current version of the pipeline can be determined.

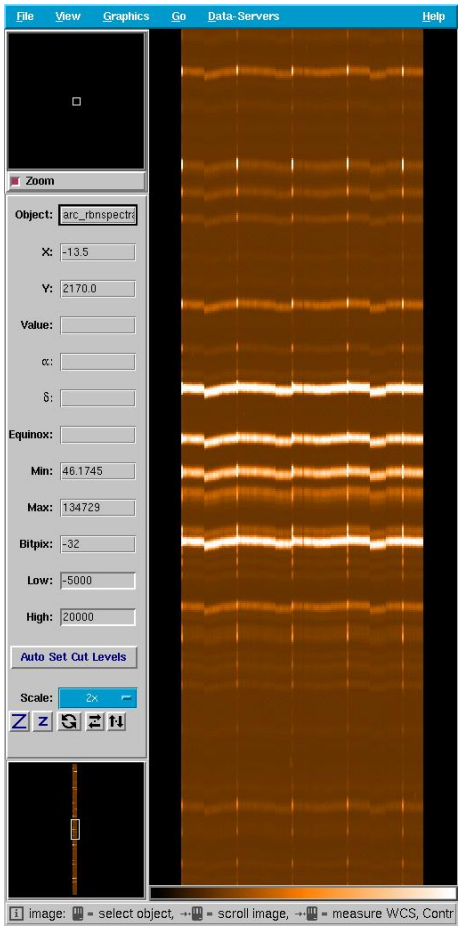


Figure 10.6: *A problematic WaveCalibration solution.*

11 IFU and ARGUS image reconstruction

In the case of 3D spectroscopic observations with IFU or ARGUS, a data cube containing the spatial information for each wavelength bin is generated, see figure 11.1. An error cube is also generated as shown below.

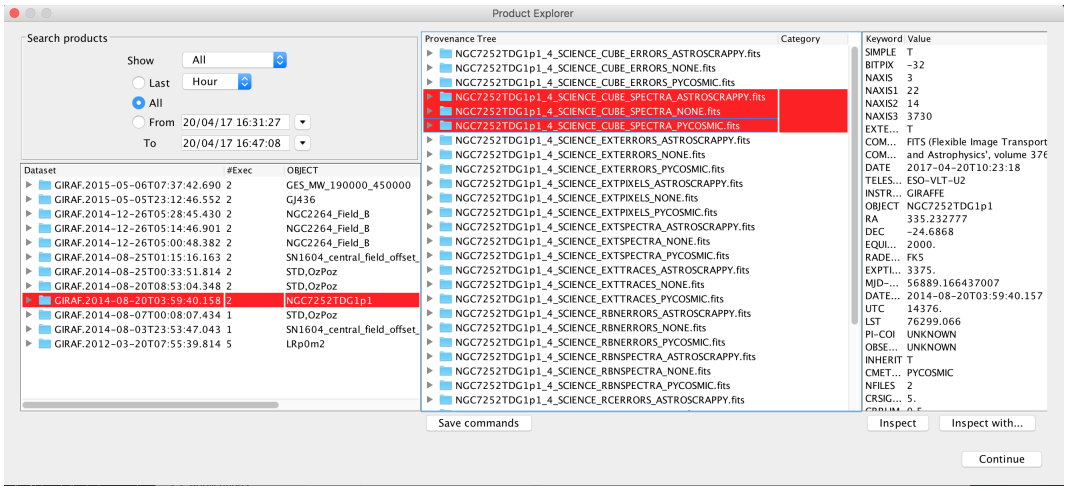


Figure 11.1: *The Product Explorer window with the CUBE products of the ARGUS science dataset highlighted in red..*

11.1 IFU and ARGUS image visualisation

At the moment, there is no dedicated tool for GIRAFFE data cubes. A nice tool originally developed for SINFONI, is `QfitsView` written by Thomas Ott. You should obtain it from:

http://www.mpe.mpg.de/~ott/QFitsView/

where Linux, Mac OS and Windows binary versions, as well as the source code is available[31]

`QFitsView` has many nice functionalities to analyze and visualise your data. The new version of QFitsView will read ARGUS and IFU mode cubes straight 'out of the box' without any need for changes to the GIRAFFE headers. The cube files have "SCIENCE_CUBE_SPECTRA" or "STD_CUBE_SPECTRA" in their file names.

An alternative is `gaia` which is available at:

http://star-www.dur.ac.uk/~pdraper/gaia/gaia.html

To visualise an ARGUS or IFU image cube, you need to wait till the end of processing all datasets is completed and the Product Explorer is displayed. Once it is, navigate within the Product Explorer

[31] `QfitsView` is included in ESO's `SciSoft` package, but, as at the time of writing, the current release (8) of `SciSoft` is based on Fedora 20 and is thus rather old now, and since the `QFitsView` version included is also rather old, we do not recommend using `QFitsView` from `SciSoft 8`.

one of the CUBE_SPECTRA files, then click on the `Inspect with...` button[32], then enter the full path to QFitsView[33] in the 'Input' window, and then click the `Ok` button, see figure 11.2.

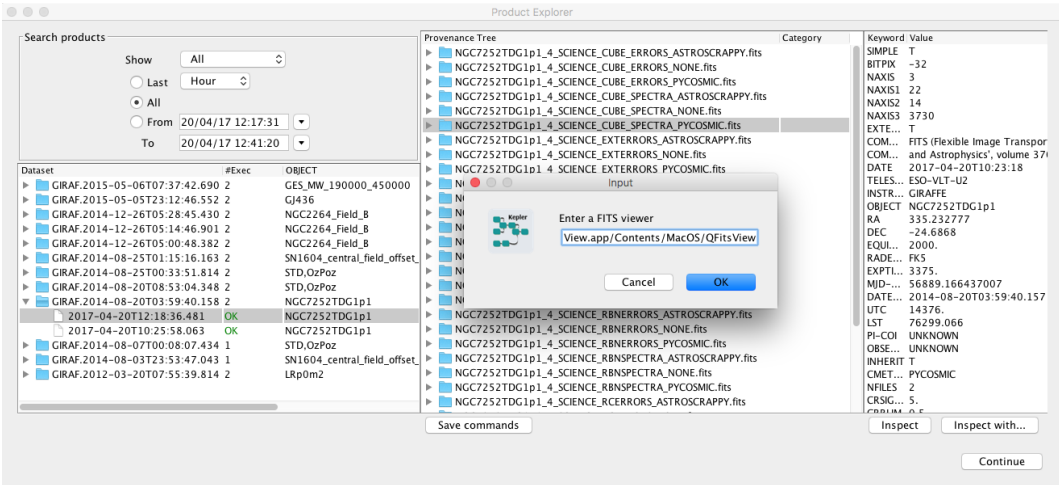


Figure 11.2: *Launching* `QFitsView` *from the Product Explorer window.*

When `QFitsView` opens it will display a window where you can choose which of the FITS file extensions you wish to open, you can click the `Read All` button, but for ARGUS products we recommend you click the `Primary` button while for IFU products re recommend that you examine each extension independently[34], see figure 11.3.

Once the Primary extension of the ARGUS product is loaded into `QFitsView` it should look something like figure 11.4.

You can then explore the cube using the slider near the top of the window, or the up and down buttons or the direct bin entry field just to the above the slider, and when you move the mouse over the image, the spectrum of the underlying pixel is then displayed in the window below. The bin displayed by defualt on startup for the ARGUS cube constructed from the demodata dataset is rather featureless, just background sky see figure 11.4, but if you navigate to bin 1156, you will see a nice bright emission feature of the galaxy, see figure 11.5. You can then move the mouse of the image to see the spectrum of the pixel under the mouse in the spectrum display subwindow. As you move over the brigth feature you will see the bright emission line apear in the spectrum.

You can mouse click-drag in the spectrum display subwindow to zoom in on that spectral region of interest. Figure 11.6 shows a zoom in the 660-675nm spectral range while the mouse is over the bright feature.

Figure 11.7 shows the IFU mode standard star cube loaded into `QFitsView`. In this case `QFitsView` successfully loaded all the extensions, however under some circumstances it crashes when one clicks the `Read All` button, in this case load the IFU units one at a time, use the `Reload` option in the

[32] You may need to increase the width of the Product Explorer window to reveal the `Inspect with...` button.
[33] On a Mac OS system this is probably something like `/Applications/QFitsView.app/Contents/MacOS/QFitsView`.
[34] In some cases `QFitsView` crashes when one attempts to read all IFU extensions.

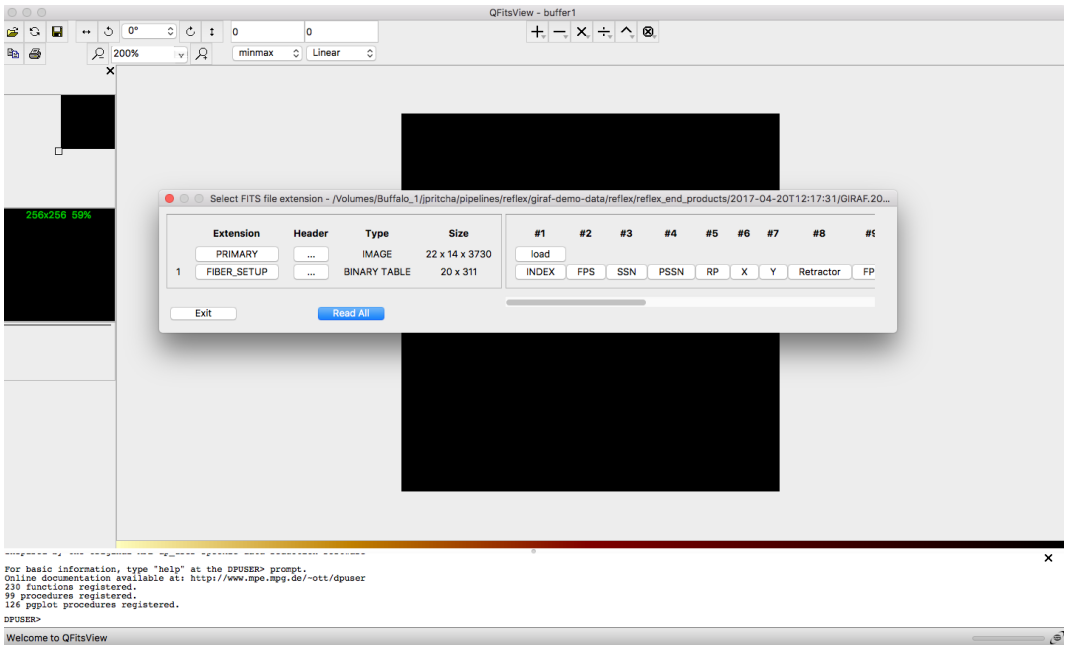


Figure 11.3: `QFitsView`*: select the FITS extension to inspect. For ARGUS CUBE products click the* `Primary` *button.*

`File` menu to navigate between extensions.

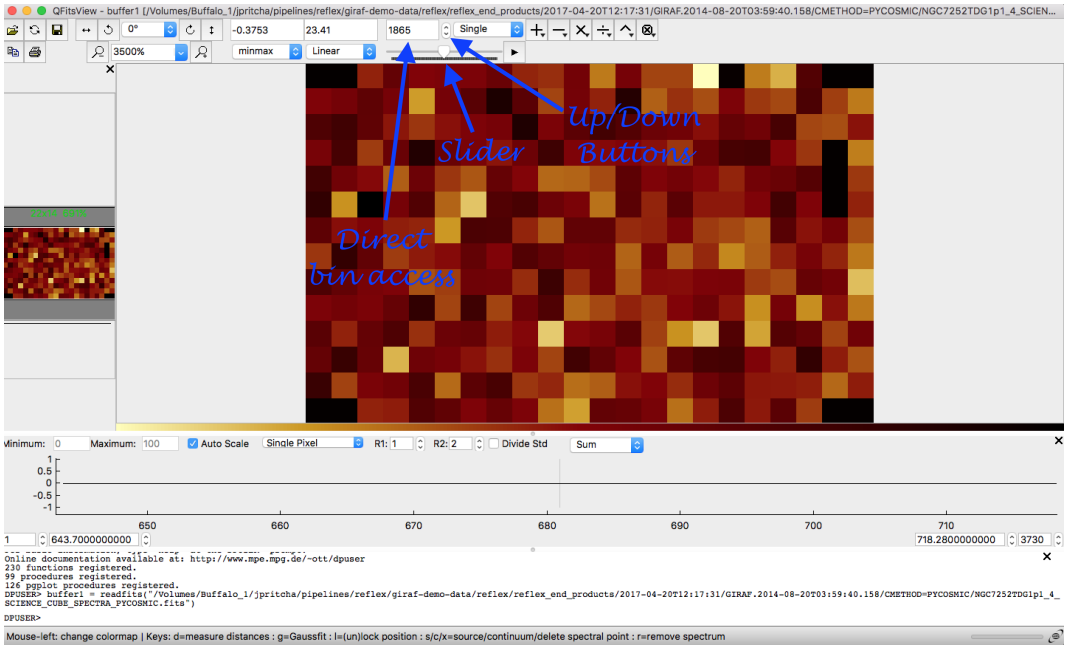


Figure 11.4: *The ARGUS science cube product loaded into* `QFitsView`.

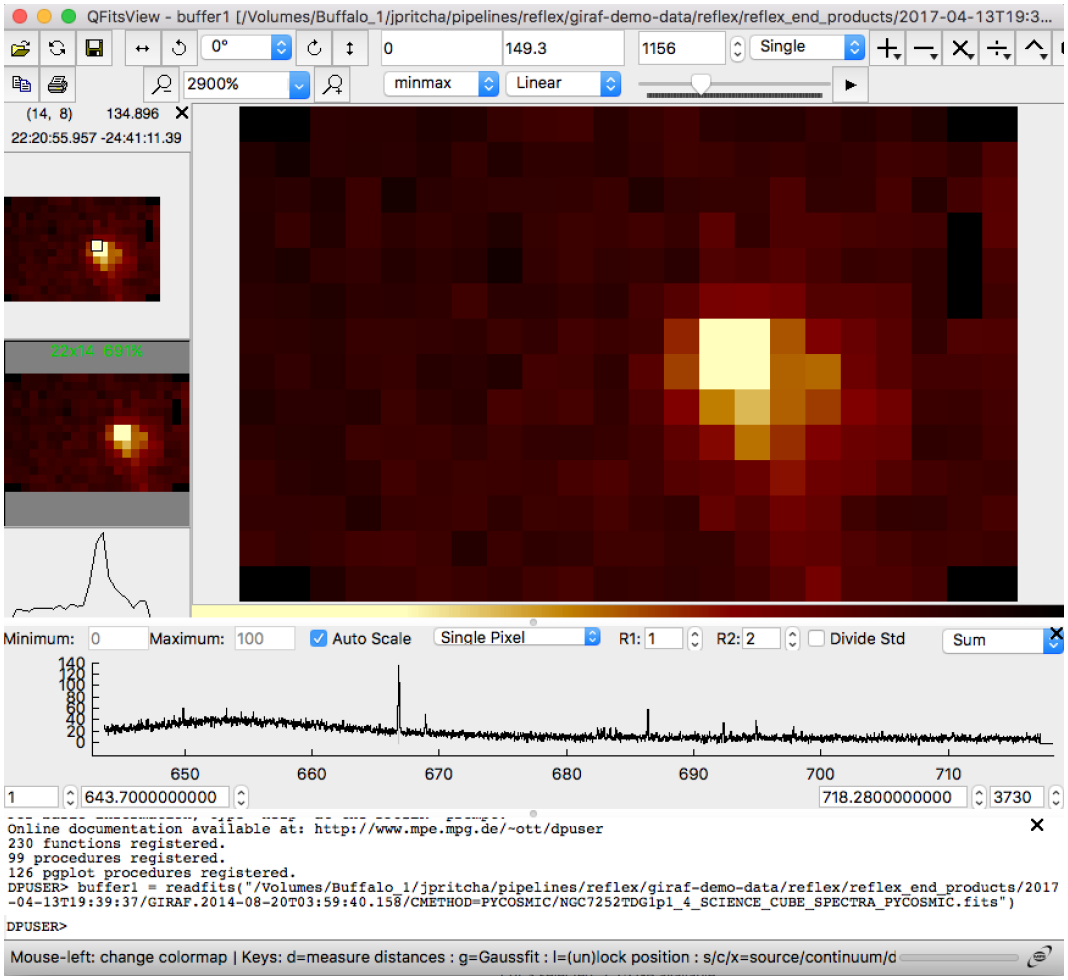


Figure 11.5: *The ARGUS science cube product loaded into* `QFitsView` *displaying bin 1156. Use the slider and/or the up and down buttons and/or the direct input field to set the bin to 1156. Now we see nice emission fromthe galaxy. Mouse click-drag in the spectrum window to zoom in on the spectral feature, see figure 11.6.*

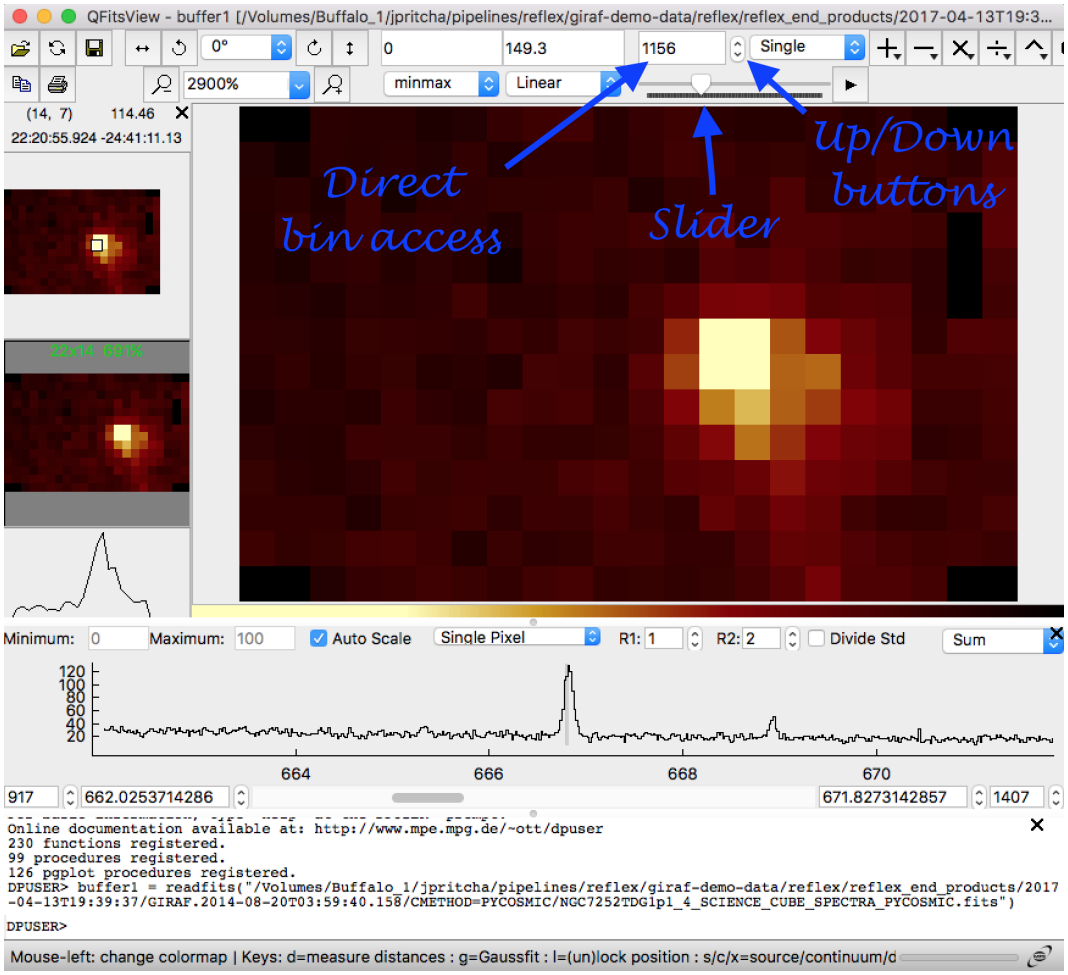


Figure 11.6: *The ARGUS science cube product loaded into* `QFitsView` *displaying bin 1156 with the spectral dispaly subwindow zoomed in on the 660-675nm wavelength range.*

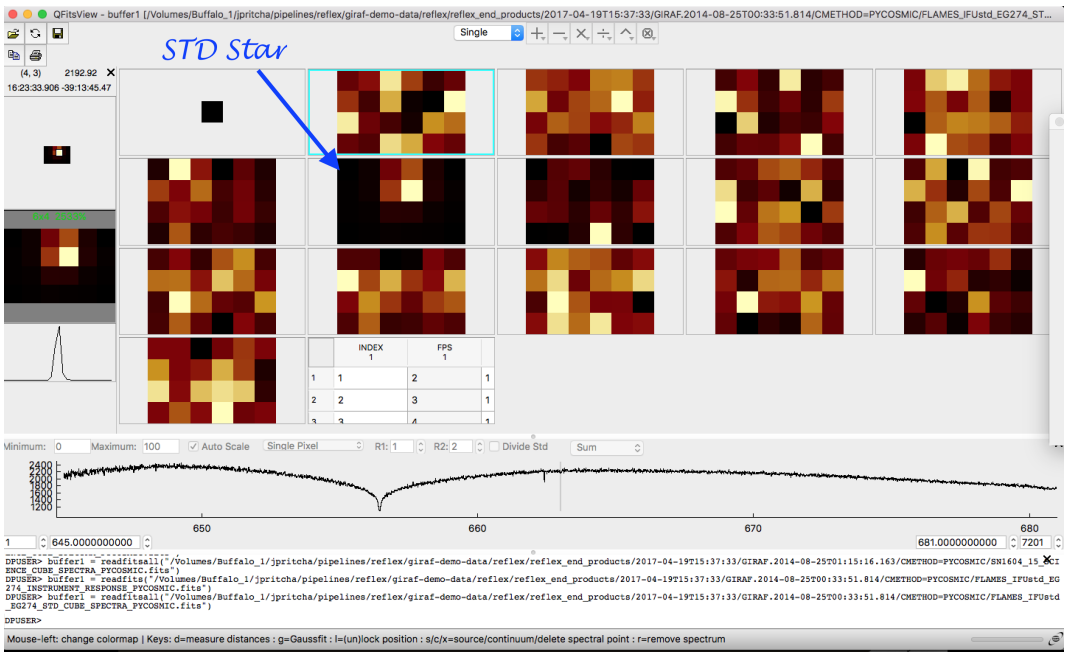


Figure 11.7: *The IFU STD cube product loaded into* `QFitsView` *displaying all 15 enabled IFU units. The standard star was in the 6th IFU unit, the spectral display subwidow shows the spectrum of the brightest pixel from that IFU unit.*

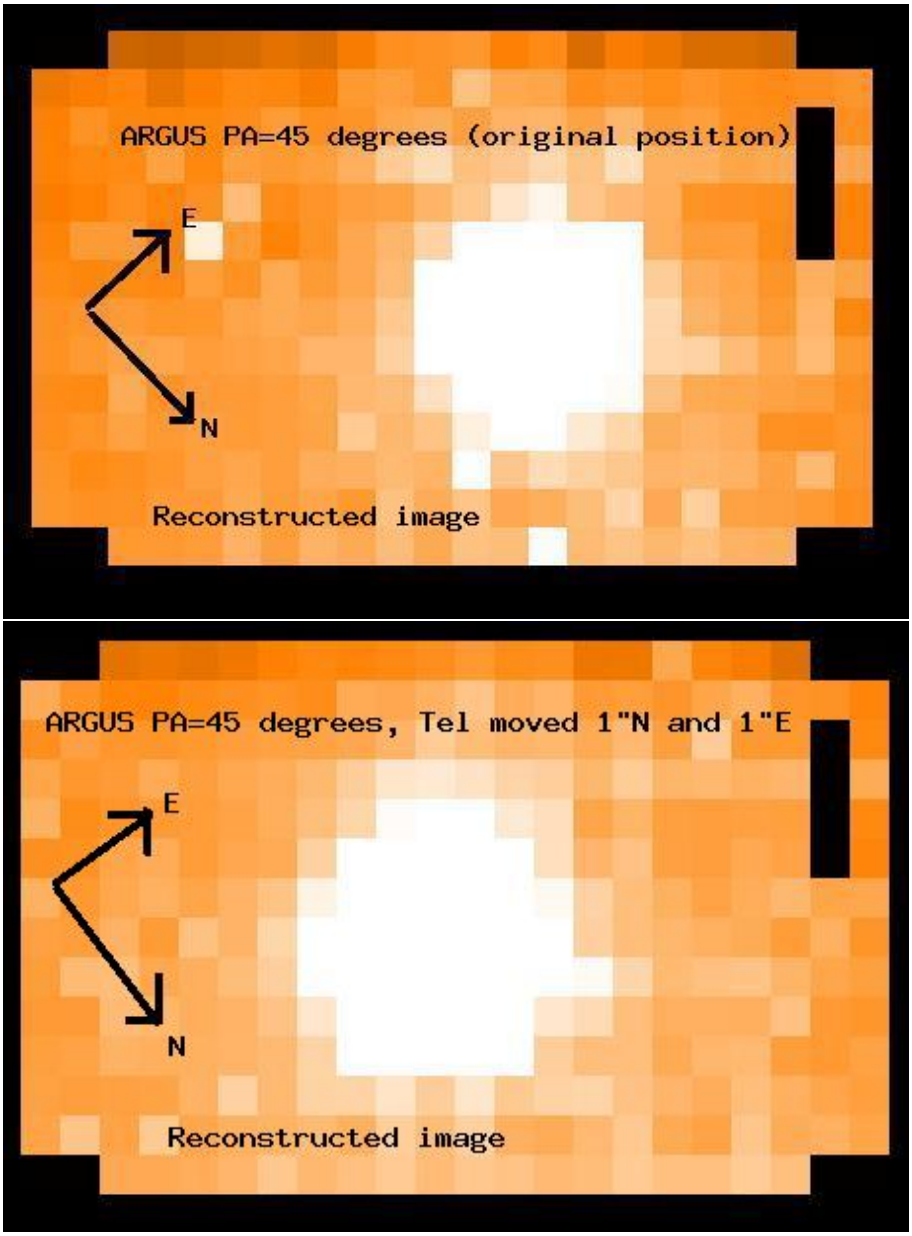


Figure 11.8: ARGUS reconstructed image with Position angle in the acquisition set to +45 degrees. Top panel: Original pointing. Bottom panel: Telescope moved by 1.0 arcseconds North and 1.0 arcseconds East i.e. the object moves 1.0 arcseconds South and 1.0 arcseconds West on ARGUS.

11.2 Plate dependent IFU response

The IFUs on the two plates have somewhat different responses. Fig 11.9 shows a sky spectrum taken on plate 1 and on plate 2.

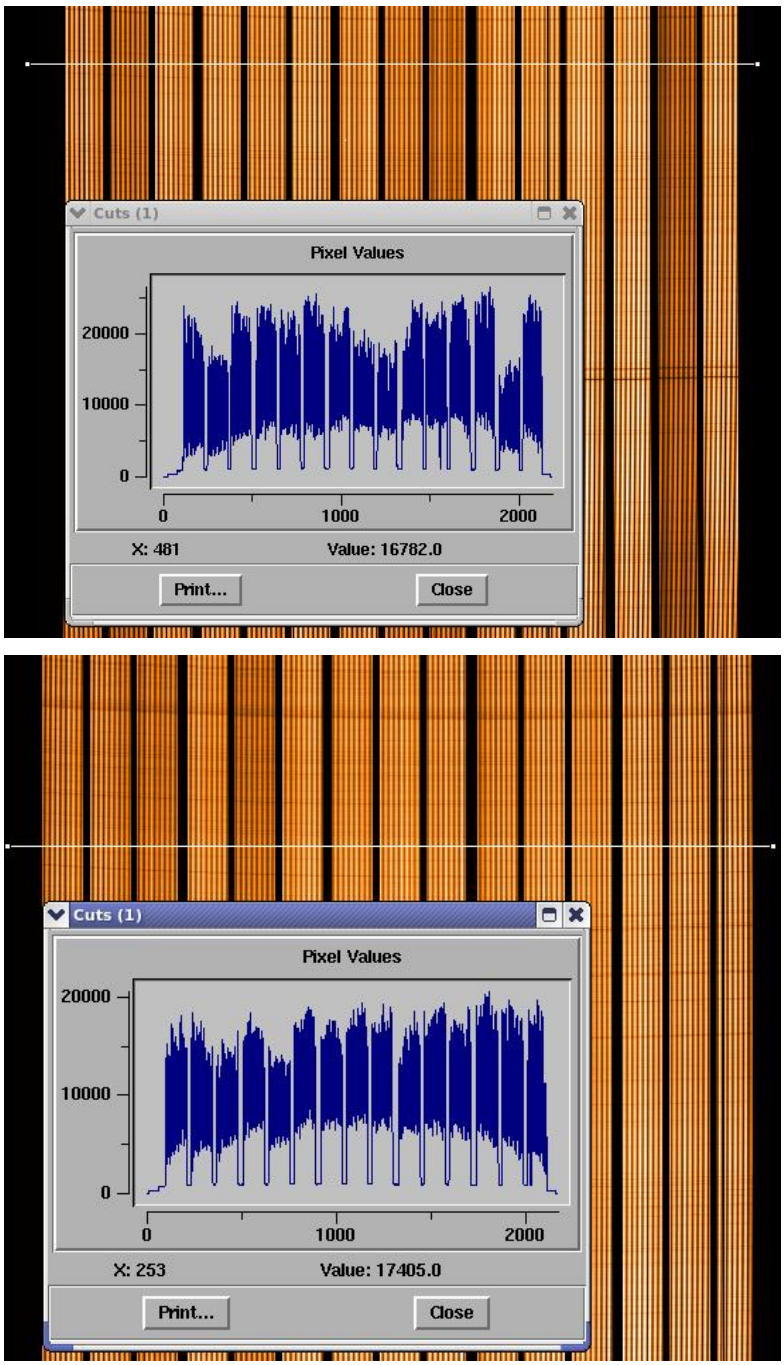


Figure 11.9: Raw GIRAFFE IFU images of the solar spectrum on plate 1 (top) and plate 2 (bottom). Variations in the IFU responses are clear.

[1] G. Battaglia, M. Irwin, E. Tolstoy, V. Hill, A. Helmi, B. Letarte, and P. Jablonka. Analysis and calibration of CaII triplet spectroscopy of red giant branch stars from VLT/FLAMES observations. *MNRAS*, 383:183–199, January 2008.

[2] J.Smoker C.Melo. *GIRAFFE data redcution cookbook*. ESO/PSO. VLT-MAN-ESO-13700-4034.

[3] ESO/LPO/PSO. *FLAMES User Manual*. VLT-MAN-ESO-13700-2994.

[4] ESO/SDD/DFS. *GIRAFFE Pipeline User Manual*. VLT-MAN-ESO-19500-3883.

[5] B. Husemann, S. Kamann, C. Sandin, S. F. Sánchez, R. García-Benito, and D. Mast. PyCosmic: a robust method to detect cosmics in CALIFA and other fiber-fed integral-field spectroscopy datasets. *A&A*, 545:A137, September 2012.

[6] A. Koch, E. K. Grebel, R. F. G. Wyse, J. T. Kleyna, M. I. Wilkinson, D. R. Harbeck, G. F. Gilmore, and N. W. Evans. Complexity on Small Scales: The Metallicity Distribution of the Carina Dwarf Spheroidal Galaxy. *AJ*, 131:895–911, February 2006.

[7] A. Koch, J. T. Kleyna, M. I. Wilkinson, E. K. Grebel, G. F. Gilmore, N. W. Evans, R. F. G. Wyse, and D. R. Harbeck. Stellar Kinematics in the Remote Leo II Dwarf Spheroidal Galaxy-Another Brick in the Wall. *AJ*, 134:566–578, August 2007.

[8] Forchì V. *Reflex User's Manual*. ESO/SCS/PPS, *http://www.eso.org/sci/software/esoreflex/*, 3.9 edition, 2018. VLT-MAN-ESO-19000-5037.

[9] P. G. van Dokkum. Cosmic-Ray Rejection by Laplacian Edge Detection. *PASP*, 113:1420–1427, November 2001.

[10] R. F. G. Wyse and G. Gilmore. Sky subtraction with fibres. *MNRAS*, 257:1–10, July 1992.