NHSC/PACS Web Tutorials

Running the PACS Spectrometer pipeline for unchopped line mode

PACS-303

Level 0 to 1 processing

Prepared by Dario Fadda
September 2012

Introduction

This tutorial will guide you through the interactive spectrometer pipeline from loading raw data into HIPE to obtain calibrated data with astrometry in the case of unchopped line range scan mode. This pipeline can reduce data taken in three different modes: (i) unchopped line standard, (ii) unchopped bright line, (iii) wavelength-switching (now obsolete, but still in the archive).

Pre-requisites

The following tutorials should be read before and after this one:

- ***PACS-101****: How to use these tutorials.*
- ***PACS-102****: Accessing and storing data from the Herschel Science Archive*
- ***PACS-103****: Loading scripts*
- ***PACS-302****: Level 1 to level 2 processing*

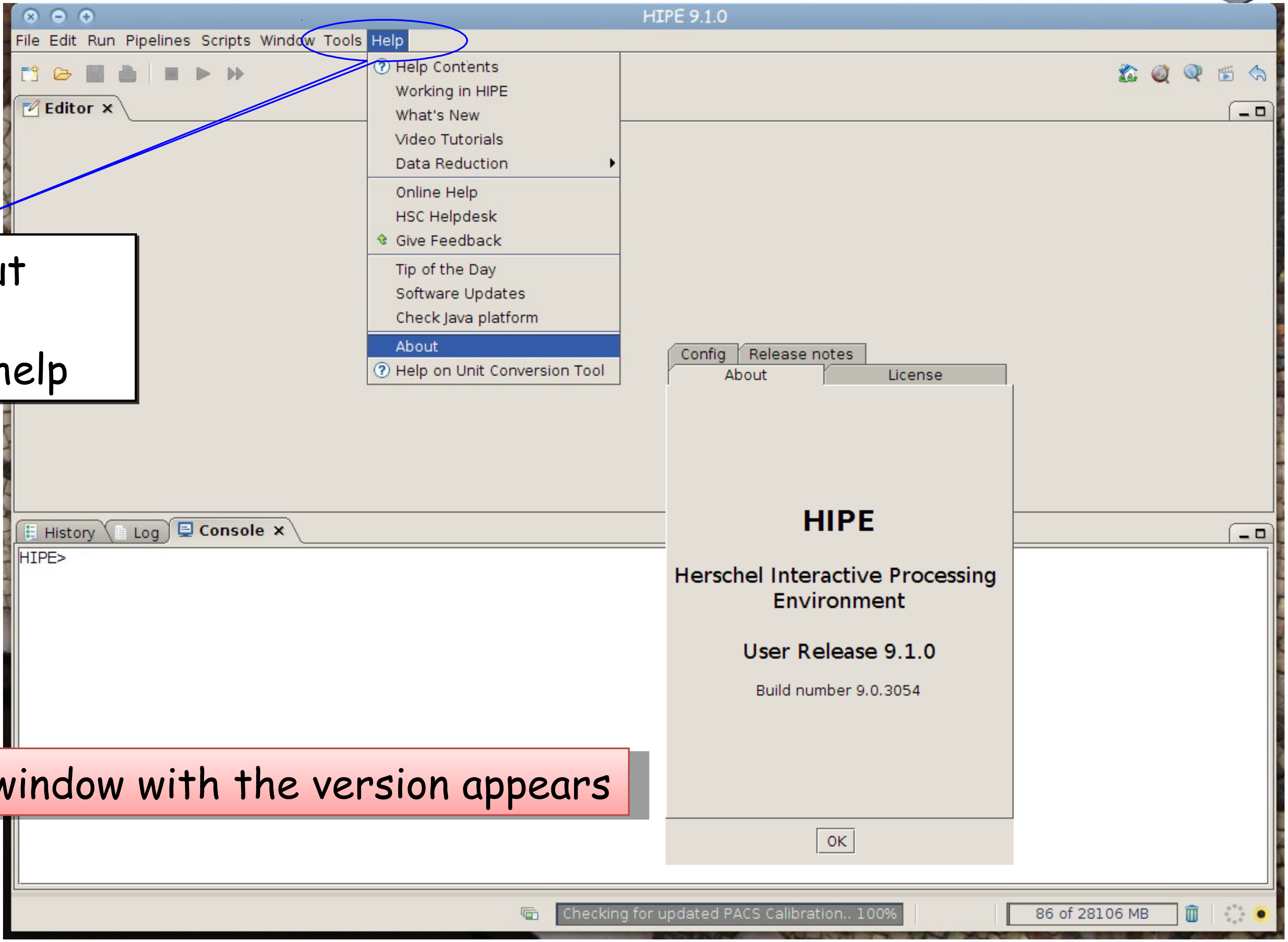
Overview

Step 1	Check HIPE version and memory
Step 2	Setup
Step 3	Run the 0 → 0.5 pipeline
Step 4	Run the 0.5 → 1 pipeline

Step 1

Check HIPE version and memory allocation

The version used for the tutorial is 9.1



HIPE>

HIPE

Herschel Interactive Processing Environment

User Release 9.1.0

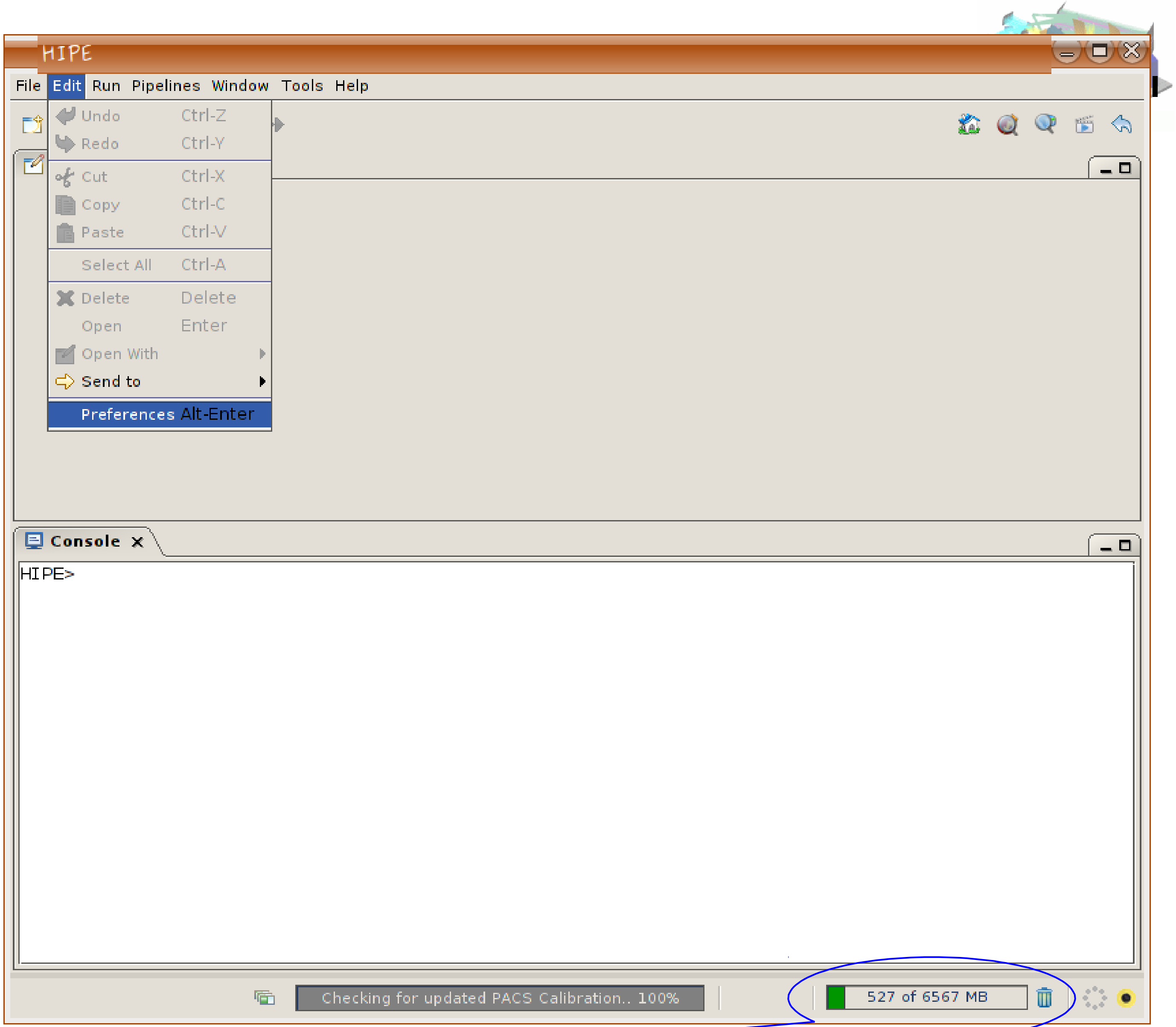
Build number 9.0.3054

Checking for updated PACS Calibration.. 100%

Select about from drop down help

A pop-out window with the version appears

To allocate memory, select preferences under edit, then ...



```
HIPE>
```

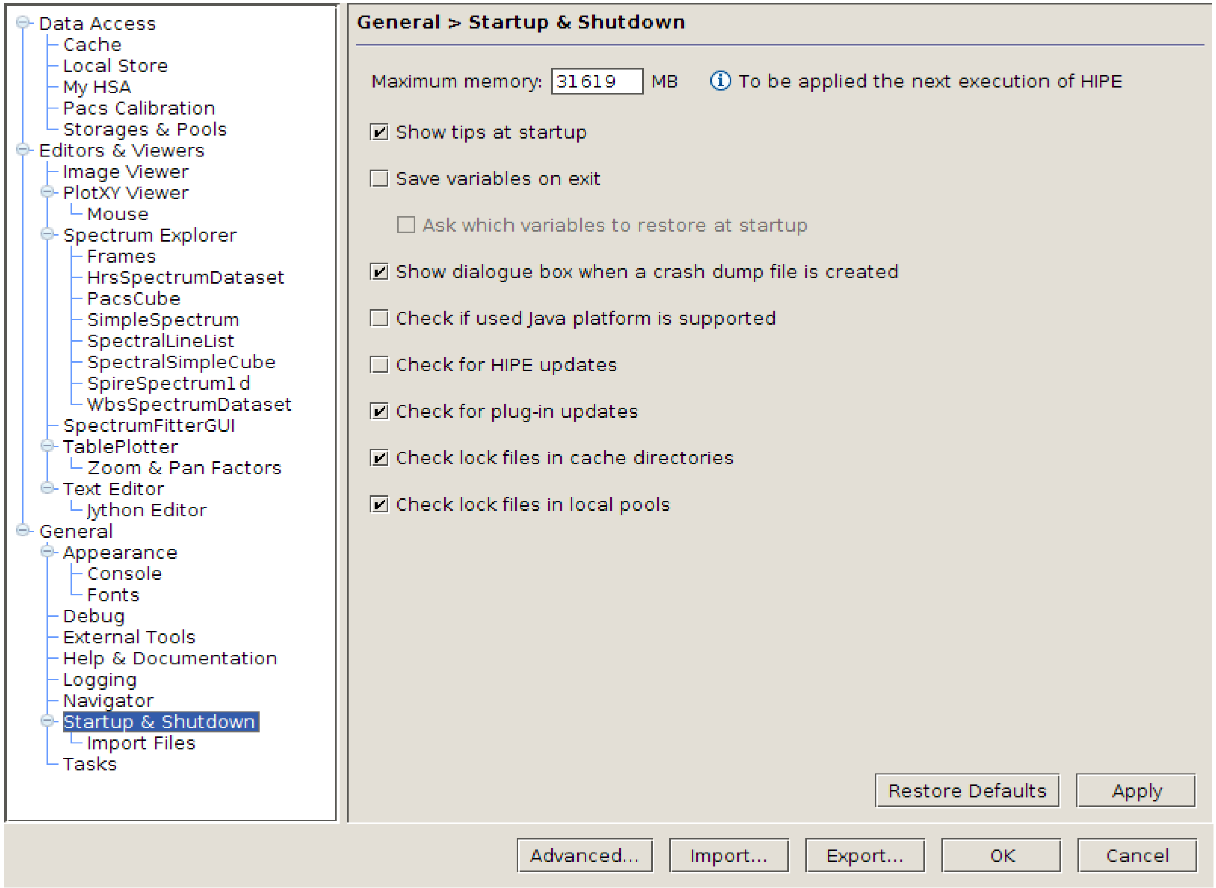
Checking for updated PACS Calibration.. 100%

527 of 6567 MB

Memory used and available

nhsc.ipac.caltech.edu/helpdesk

Then click on Startup & Shutdown and change the amount of memory



General > Startup & Shutdown

Maximum memory: 31619 MB ⓘ To be applied the next execution of HIPE

☑ Show tips at startup

☐ Save variables on exit

☐ Ask which variables to restore at startup

☑ Show dialogue box when a crash dump file is created

☐ Check if used Java platform is supported

☐ Check for HIPE updates

☑ Check for plug-in updates

☑ Check lock files in cache directories

☑ Check lock files in local pools

The allocated memory should be smaller than the total RAM of your computer. You have to exit and start a new session to use the new amount of memory.

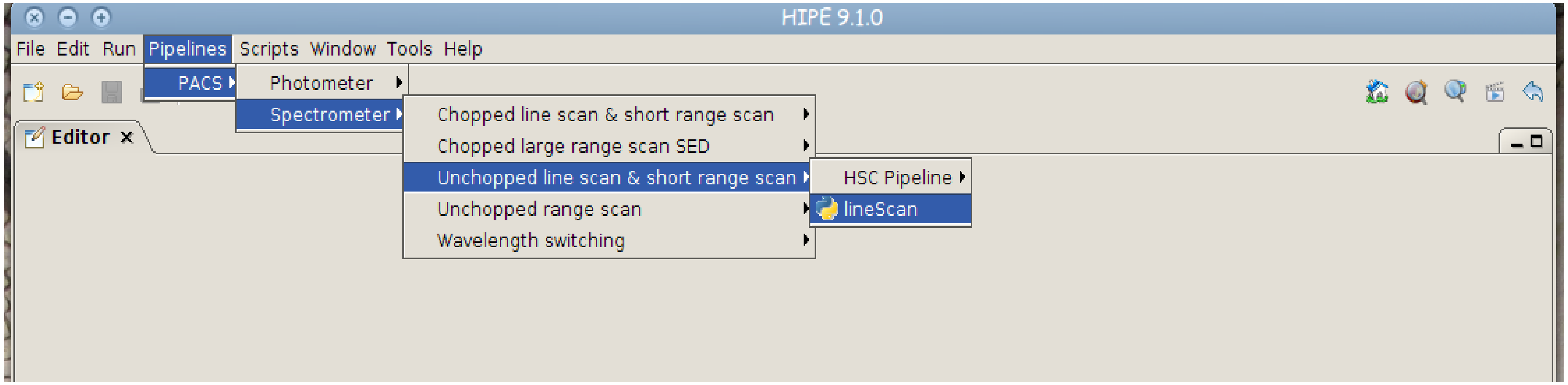
Step 2

Setup

Load pipeline script, load observation, check data, and select the camera

Loading the script

The script used in this tutorial corresponds to the script available directly from the distribution.

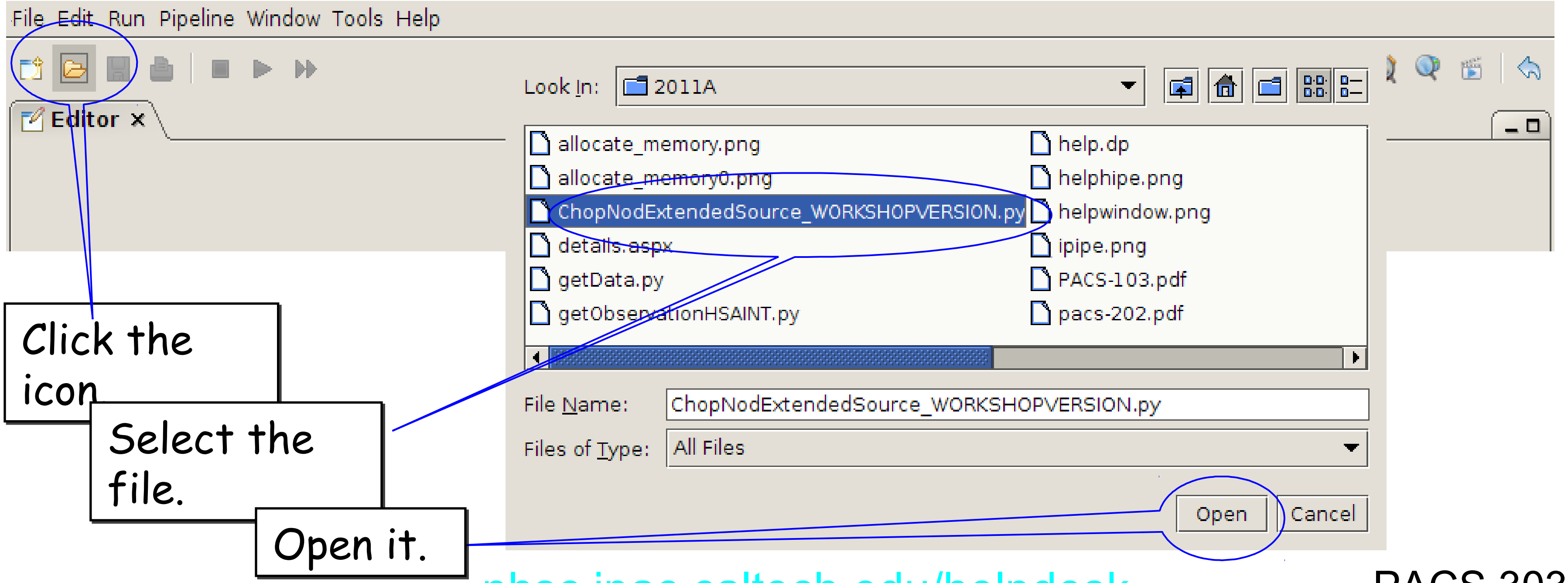

HIPE 9.1.0
File Edit Run Pipelines Scripts Window Tools Help
PACS
Photometer
Spectrometer
Chopped line scan & short range scan
Chopped large range scan SED
Unchopped line scan & short range scan
Unchopped range scan
Wavelength switching
HSC Pipeline
lineScan
Editor

In the case you were using a modified script, you should first load it from the directory where it resides.

nhsc.ipac.caltech.edu/helpdesk

Loading the script

To load a custom script into an HIPE session, just click on the loading icon as shown in the figure. The search the location where you put the file using the pop-up window and finally load it into the session.



Look In: 2011A

- allocate_memory.png
- allocate_memory0.png
- ChopNodExtendedSource_WORKSHOPVERSION.py
- details.aspx
- getData.py
- getObservationHSAINT.py
- help.dp
- helphipe.png
- helpwindow.png
- ipipe.png
- PACS-103.pdf
- pacs-202.pdf

File Name: ChopNodExtendedSource_WORKSHOPVERSION.py

Files of Type: All Files

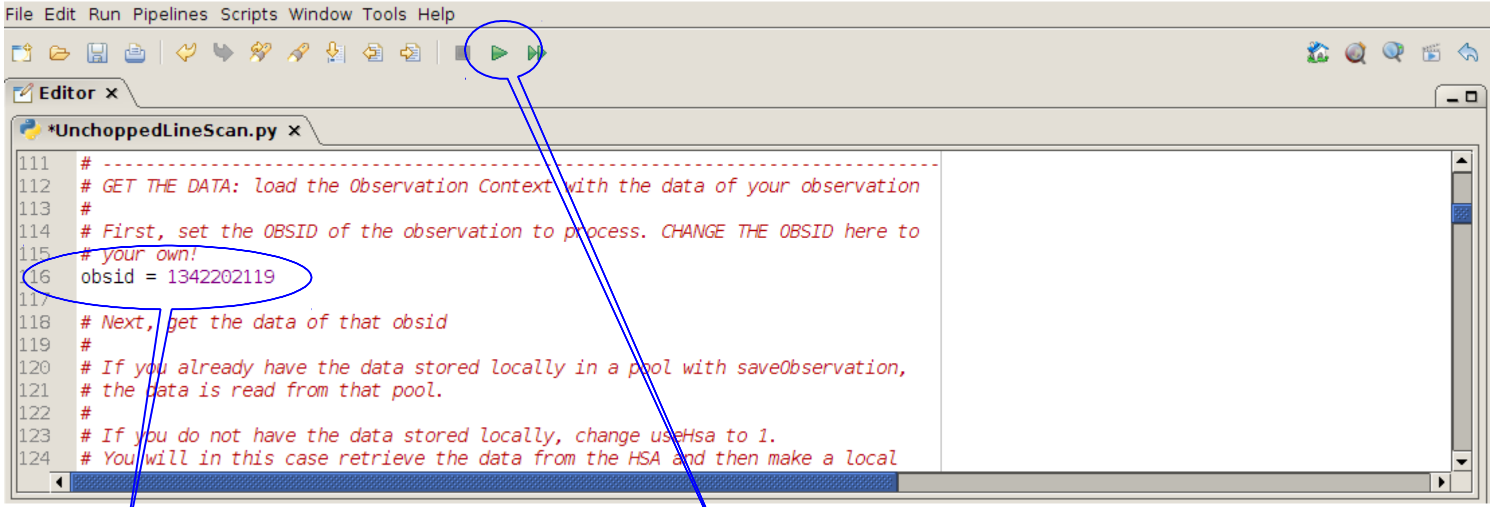
Click the icon

Select the file.

Open it.

Loading the observation

Once the file is loaded, one can simply step through the lines to execute it one by one. In this tutorial, we will explain how to modify some lines to explore different observations and lines and to check the results of the main operations on the data. The first thing to do is loading the OBSID relative to the observation chosen. In the case of this tutorial, the observations has been already saved into a pool which has to be put into your ~/.hcss/lstore directory which is created once installing HIPE.



```
# ------------------------------------------------------------------------------------
# GET THE DATA: load the Observation Context with the data of your observation
#
# First, set the OBSID of the observation to process. CHANGE THE OBSID here to
# your own!
obsid = 1342202119

# Next, get the data of that obsid
#
# If you already have the data stored locally in a pool with saveObservation,
# the data is read from that pool.
#
# If you do not have the data stored locally, change useHsa to 1.
# You will in this case retrieve the data from the HSA and then make a local
```

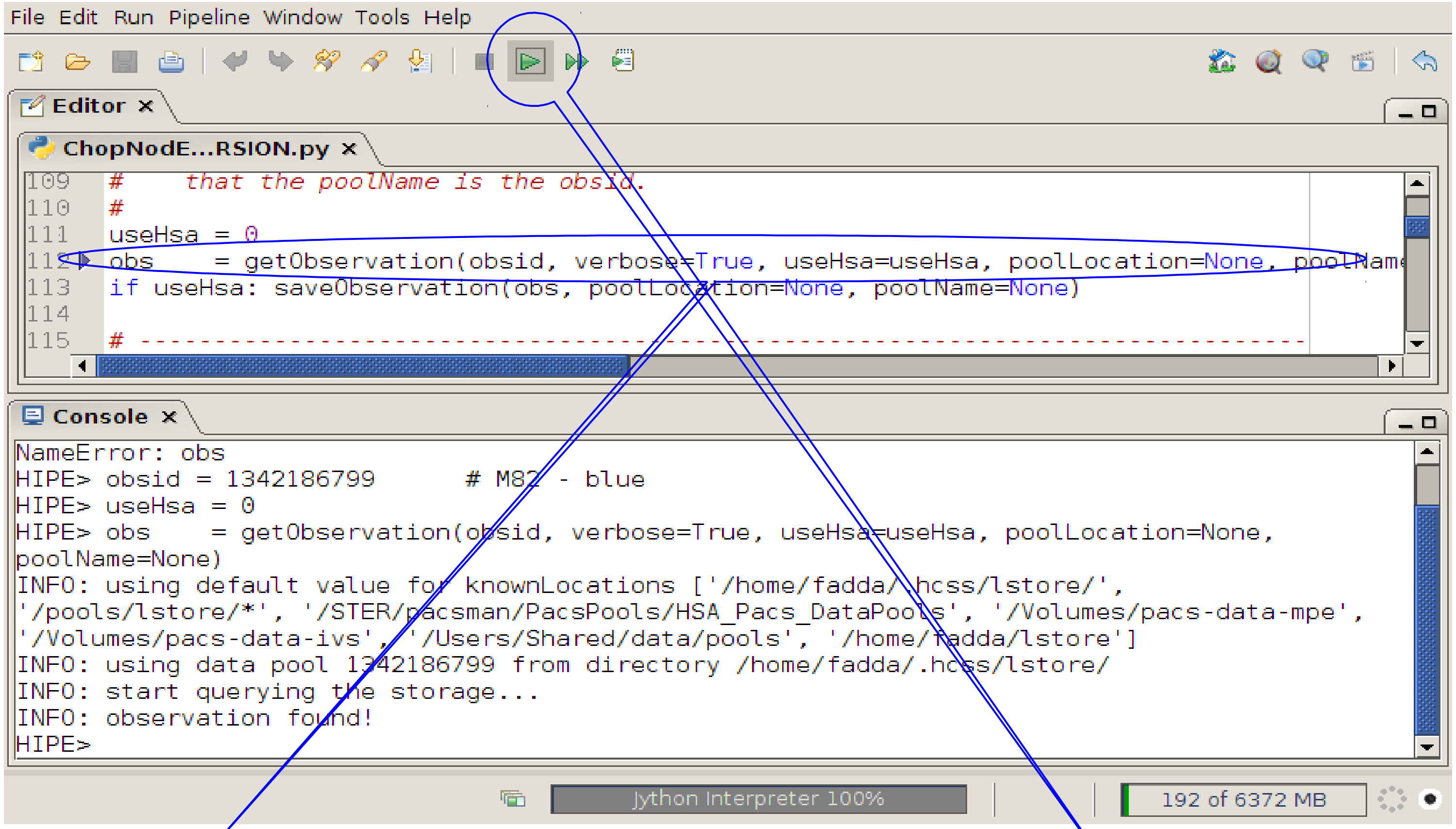
Modify this line and click on it

Hit the arrow

Loading the observation

NHSC PACS Web Tutorial

Next step, we load the observational context (a structure containing all the observational data, information about them and calibration data).



```
#     that the poolName is the obsid.
#
useHsa = 0
obs     = getObservation(obsid, verbose=True, useHsa=useHsa, poolLocation=None, poolNam
if useHsa: saveObservation(obs, poolLocation=None, poolName=None)

# --------------------------------------------------------------------------------
```

```
NameError: obs
HIPE> obsid = 1342186799      # M82 - blue
HIPE> useHsa = 0
HIPE> obs     = getObservation(obsid, verbose=True, useHsa=useHsa, poolLocation=None,
poolName=None)
INFO: using default value for knownLocations ['/home/fadda/.hcss/lstore/',
'/pools/lstore/*', '/STER/pacsman/PacsPools/HSA_Pacs_DataPools', '/Volumes/pacs-data-mpe',
'/Volumes/pacs-data-ivs', '/Users/Shared/data/pools', '/home/fadda/lstore']
INFO: using data pool 1342186799 from directory /home/fadda/.hcss/lstore/
INFO: start querying the storage...
INFO: observation found!
HIPE>
```

Click on this line.

Hit the arrow

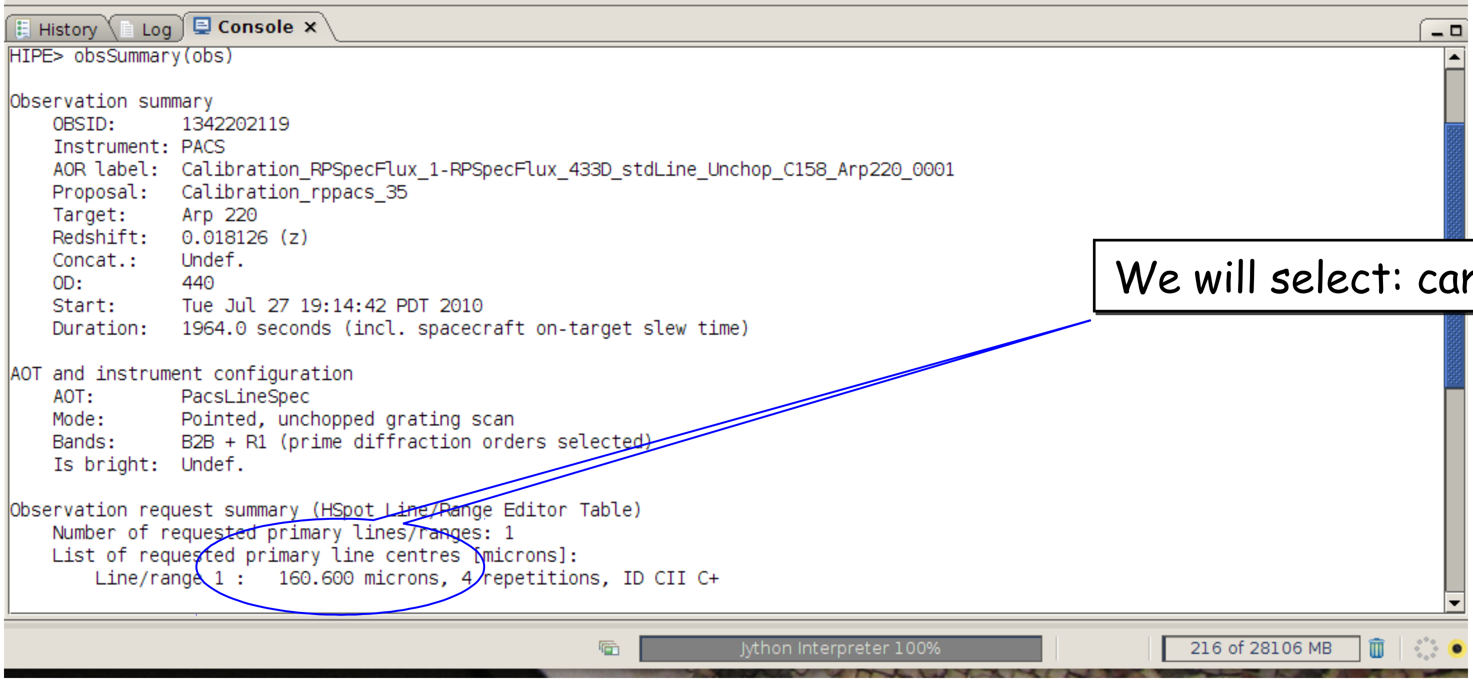
Check: observation summary

The next command to use is:

```
obsSummary(obs)
```

Although it comes later in the official pipeline, you can use it already once the observation has been loaded. This can be very instructive, especially if you don't know the lines which have been observed and you want to set the pipeline script to reduce and visualize a particular line.

Check: observation summary



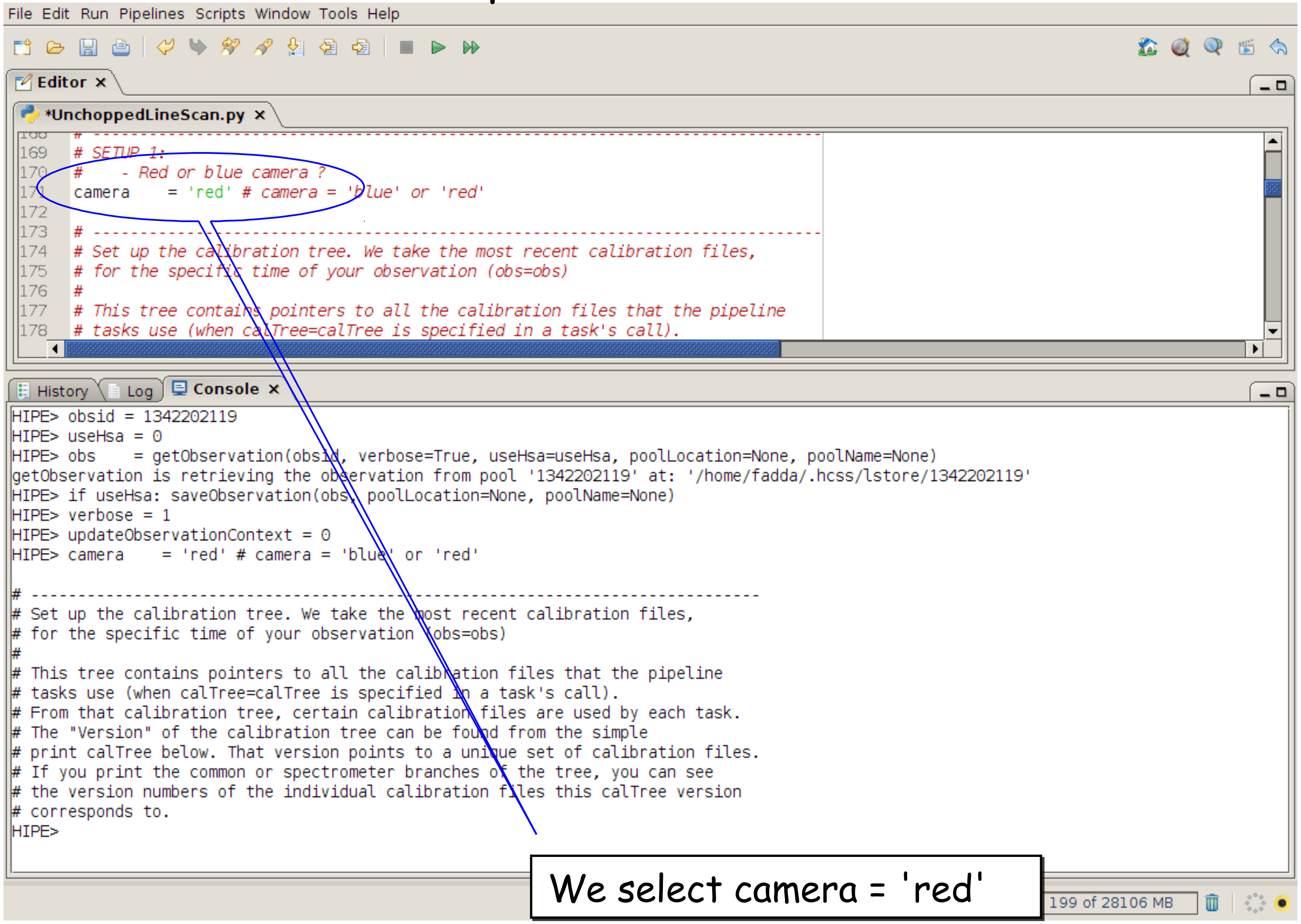
```
HIPE> obsSummary(obs)

Observation summary
    OBSID:      1342202119
    Instrument: PACS
    AOR label:  Calibration_RPSpecFlux_1-RPSpecFlux_433D_stdLine_Unchop_C158_Arp220_0001
    Proposal:   Calibration_rppacs_35
    Target:     Arp 220
    Redshift:   0.018126 (z)
    Concat.:    Undef.
    OD:         440
    Start:      Tue Jul 27 19:14:42 PDT 2010
    Duration:   1964.0 seconds (incl. spacecraft on-target slew time)

AOT and instrument configuration
    AOT:        PacsLineSpec
    Mode:       Pointed, unchopped grating scan
    Bands:      B2B + R1 (prime diffraction orders selected)
    Is bright:  Undef.

Observation request summary (HSpot Line/Range Editor Table)
    Number of requested primary lines/ranges: 1
    List of requested primary line centres [microns]:
        Line/range 1 :   160.600 microns, 4 repetitions, ID CII C+
```

We will select: camera = 'red'

Setting the camera

NHSC PACS Web Tutorial

Once we decide the line to explore, we can set the camera to blue or red.

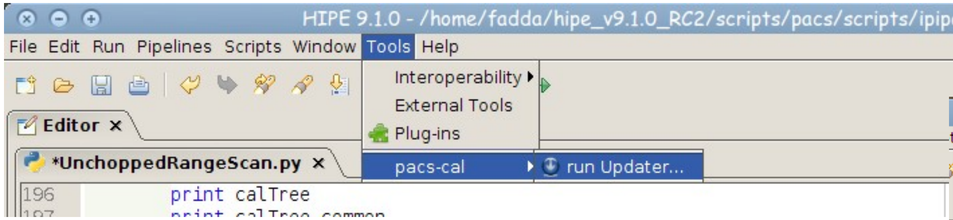


```
# ---------------------------------------------------------------------------
# SETUP 1:
#     - Red or blue camera ?
camera     = 'red' # camera = 'blue' or 'red'

# ---------------------------------------------------------------------------
# Set up the calibration tree. We take the most recent calibration files,
# for the specific time of your observation (obs=obs)
#
# This tree contains pointers to all the calibration files that the pipeline
# tasks use (when calTree=calTree is specified in a task's call).
```

```
HIPE> obsid = 1342202119
HIPE> useHsa = 0
HIPE> obs     = getObservation(obsid, verbose=True, useHsa=useHsa, poolLocation=None, poolName=None)
getObservation is retrieving the observation from pool '1342202119' at: '/home/fadda/.hcss/lstore/1342202119'
HIPE> if useHsa: saveObservation(obs, poolLocation=None, poolName=None)
HIPE> verbose = 1
HIPE> updateObservationContext = 0
HIPE> camera     = 'red' # camera = 'blue' or 'red'

# ---------------------------------------------------------------------------
# Set up the calibration tree. We take the most recent calibration files,
# for the specific time of your observation (obs=obs)
#
# This tree contains pointers to all the calibration files that the pipeline
# tasks use (when calTree=calTree is specified in a task's call).
# From that calibration tree, certain calibration files are used by each task.
# The "Version" of the calibration tree can be found from the simple
# print calTree below. That version points to a unique set of calibration files.
# If you print the common or spectrometer branches of the tree, you can see
# the version numbers of the individual calibration files this calTree version
# corresponds to.
HIPE>
```

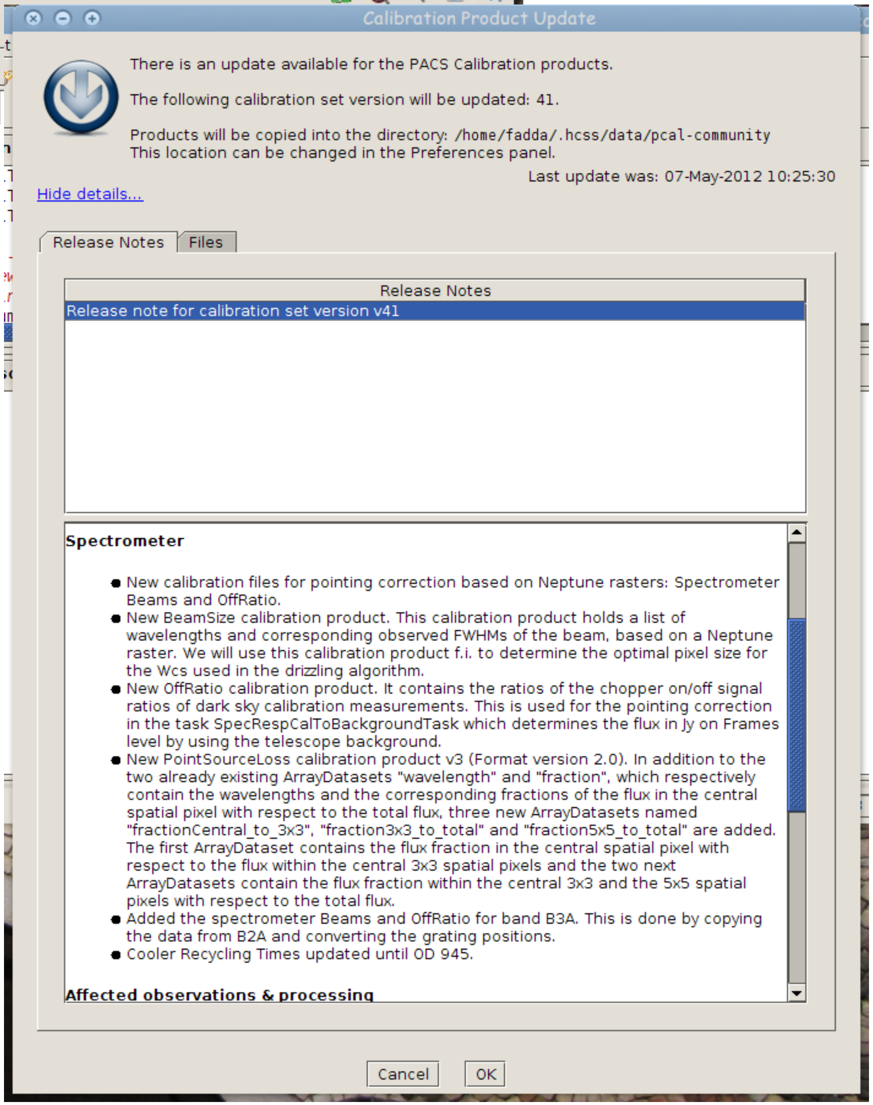
We select camera = 'red'

nhsc.ipac.caltech.edu/helpdesk


HIPE 9.1.0 - /home/fadda/hipe_v9.1.0_RC2/scripts/pacs/scripts/ipipe/spec/UnchoppedRangeScan.py
File Edit Run Pipelines Scripts Window Tools Help
Interoperability
External Tools
Plug-ins
pacs-cal
run Updater...
Editor
*UnchoppedRangeScan.py

```
        print calTree
        print calTree.common
```

We can check if the calibration is up to date, by running the pacs-cal updater. In the case a new version is installed, we can inspect the release notes. In the figure, we can see the new features implemented in the current version of calibration (41). At the time of writing, the ICC version of calibration is already 45 so, expect soon new developments ...



There is an update available for the PACS Calibration products.

The following calibration set version will be updated: 41.

Products will be copied into the directory: /home/fadda/.hcss/data/pcal-community
This location can be changed in the Preferences panel.

Last update was: 07-May-2012 10:25:30

Release Notes
Release note for calibration set version v41

Spectrometer

- New calibration files for pointing correction based on Neptune rasters: Spectrometer Beams and OffRatio.
- New BeamSize calibration product. This calibration product holds a list of wavelengths and corresponding observed FWHMs of the beam, based on a Neptune raster. We will use this calibration product f.i. to determine the optimal pixel size for the Wcs used in the drizzling algorithm.
- New OffRatio calibration product. It contains the ratios of the chopper on/off signal ratios of dark sky calibration measurements. This is used for the pointing correction in the task SpecRespCalToBackgroundTask which determines the flux in Jy on Frames level by using the telescope background.
- New PointSourceLoss calibration product v3 (Format version 2.0). In addition to the two already existing ArrayDatasets "wavelength" and "fraction", which respectively contain the wavelengths and the corresponding fractions of the flux in the central spatial pixel with respect to the total flux, three new ArrayDatasets named "fractionCentral_to_3x3", "fraction3x3_to_total" and "fraction5x5_to_total" are added. The first ArrayDataset contains the flux fraction in the central spatial pixel with respect to the flux within the central 3x3 spatial pixels and the two next ArrayDatasets contain the flux fraction within the central 3x3 and the 5x5 spatial pixels with respect to the total flux.
- Added the spectrometer Beams and OffRatio for band B3A. This is done by copying the data from B2A and converting the grating positions.
- Cooler Recycling Times updated until OD 945.

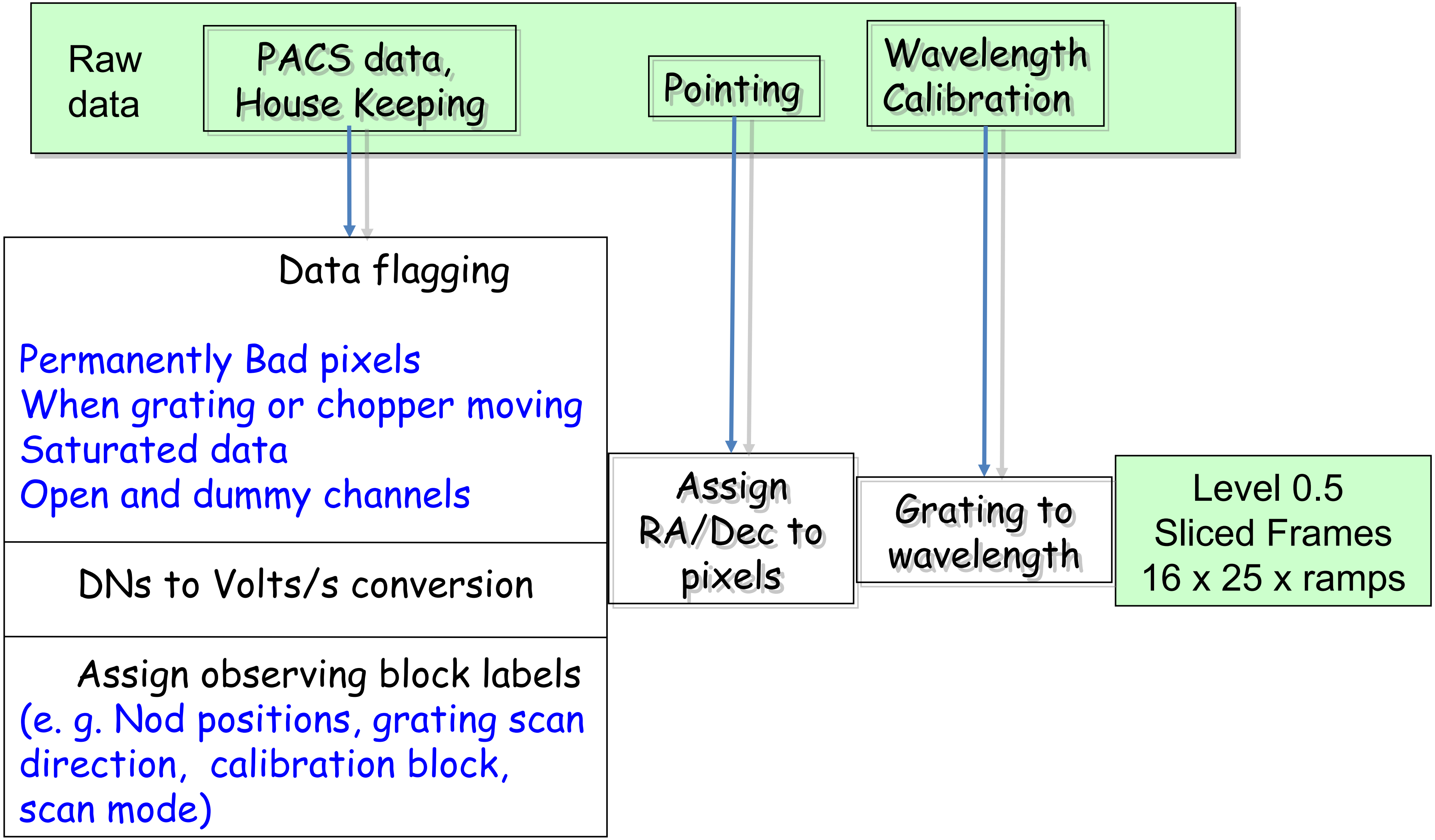
Affected observations & processing

Step 3

Run the 0 → 0.5 pipeline

Basic calibration (pointing, wavelength calibration, slicing)

Level 0 → 0.5



Raw data

PACS data, House Keeping

Pointing

Wavelength Calibration

Data flagging

Permanently Bad pixels
When grating or chopper moving
Saturated data
Open and dummy channels

DNs to Volts/s conversion

Assign observing block labels (e. g. Nod positions, grating scan direction, calibration block, scan mode)

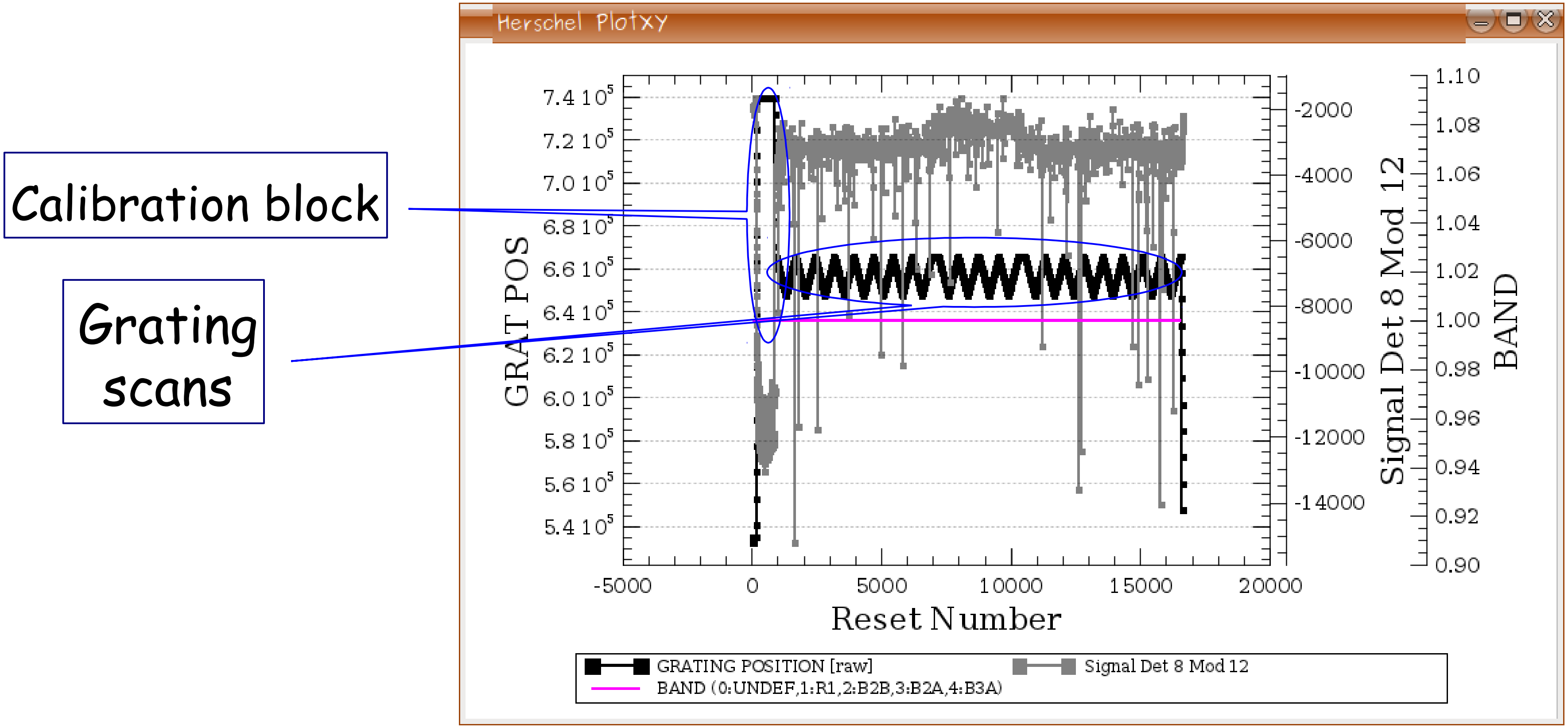
Assign RA/Dec to pixels

Grating to wavelength

Level 0.5 Sliced Frames 16 x 25 x ramps

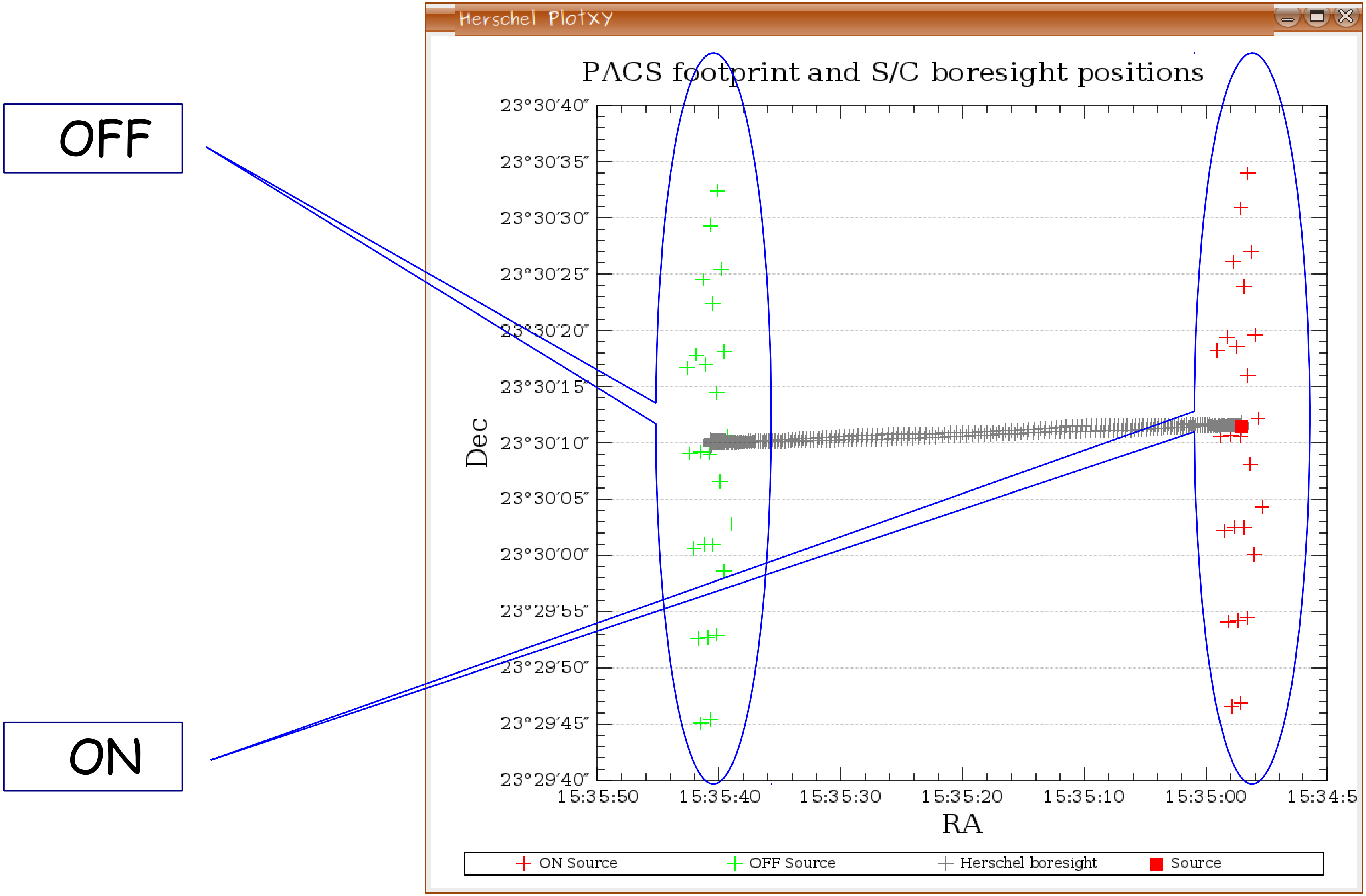
Check: level 0

From now on, we will step through the script line by line using the green arrow on the menu bar. The first step consists in extracting the 0-level products from the observation context.



In our case, after the calibration block, a line is observed in R1.

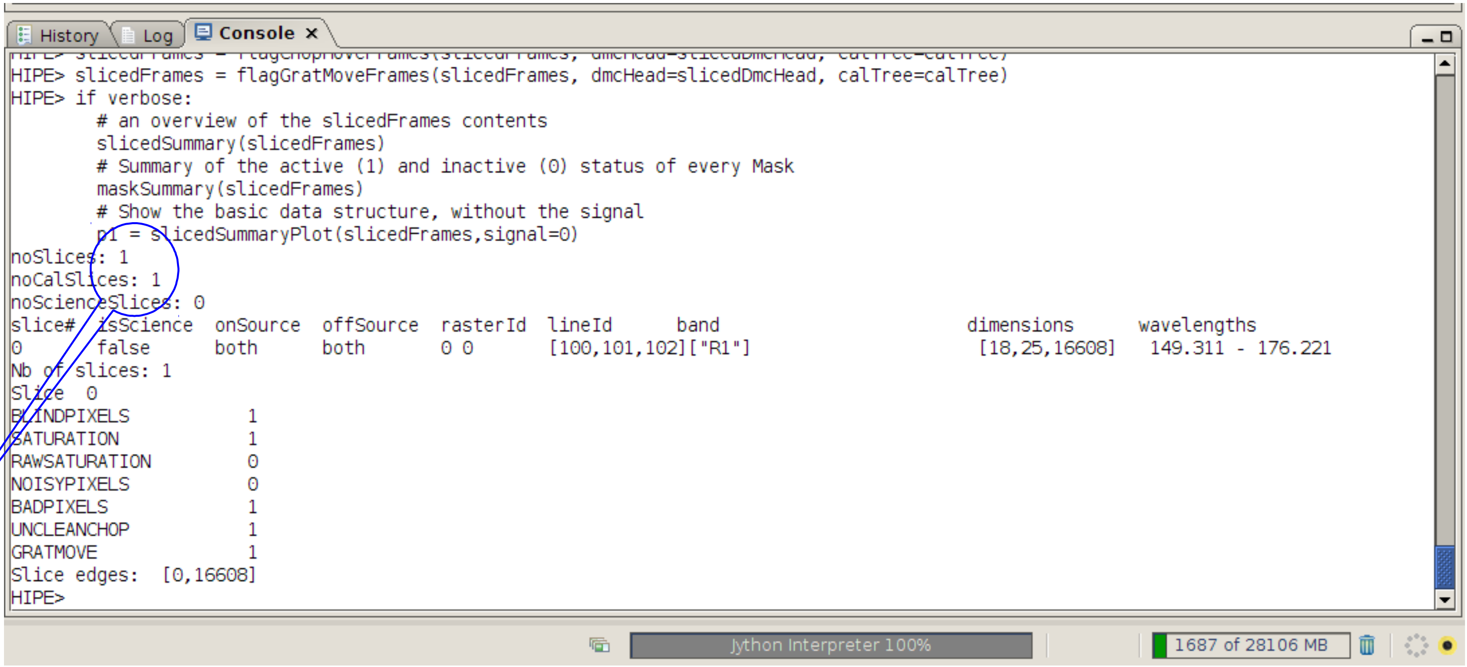
Check: footprint



OFF

ON

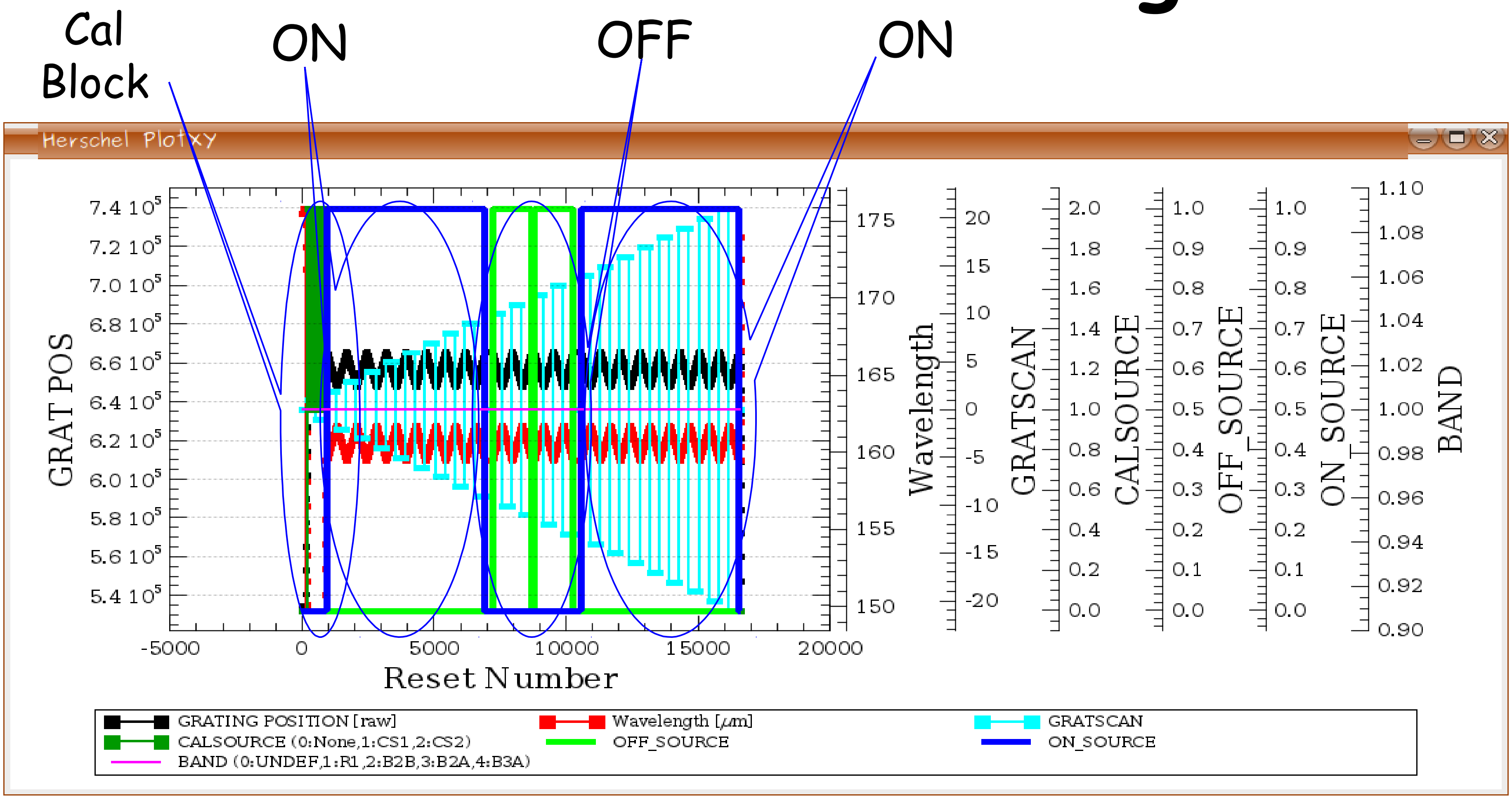
Check: before slicing



```
HIPE> slicedFrames = flagChopMoveFrames(slicedFrames, dmcHead=slicedDmcHead, calTree=calTree)
HIPE> slicedFrames = flagGratMoveFrames(slicedFrames, dmcHead=slicedDmcHead, calTree=calTree)
HIPE> if verbose:
        # an overview of the slicedFrames contents
        slicedSummary(slicedFrames)
        # Summary of the active (1) and inactive (0) status of every Mask
        maskSummary(slicedFrames)
        # Show the basic data structure, without the signal
        p1 = slicedSummaryPlot(slicedFrames,signal=0)
noSlices: 1
noCalSlices: 1
noScienceSlices: 0
slice#  isScience  onSource   offSource  rasterId   lineId      band                        dimensions      wavelengths
0       false      both       both       0 0        [100,101,102]["R1"]                      [18,25,16608]   149.311 - 176.221
Nb of slices: 1
Slice  0
BLINDPIXELS            1
SATURATION             1
RAWSATURATION          0
NOISYPIXELS            0
BADPIXELS              1
UNCLEANCHOP            1
GRATMOVE               1
Slice edges:  [0,16608]
HIPE>
```

Only 1 slice

Check: before slicing



Cal Block

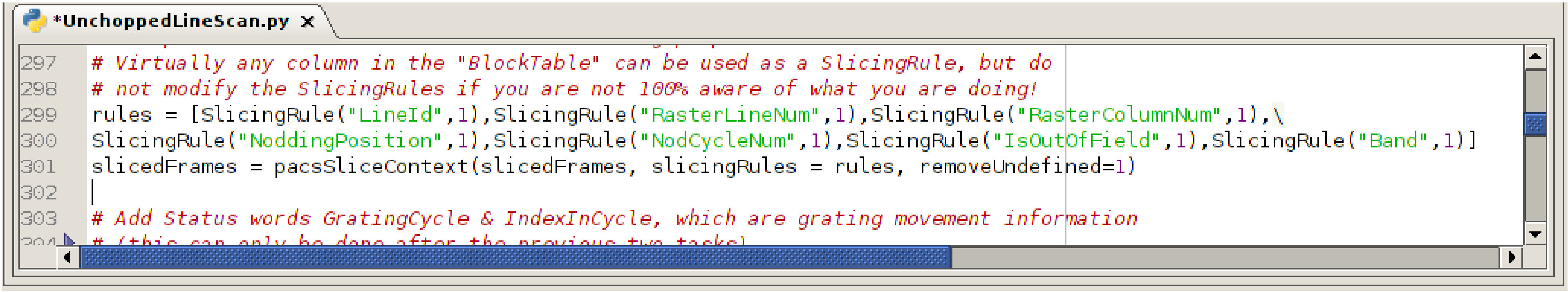
ON

OFF

ON

One line with ON, OFF, and ON source positions. Grating scans are numbered positive if upscans and negative if downscans.

Slicing

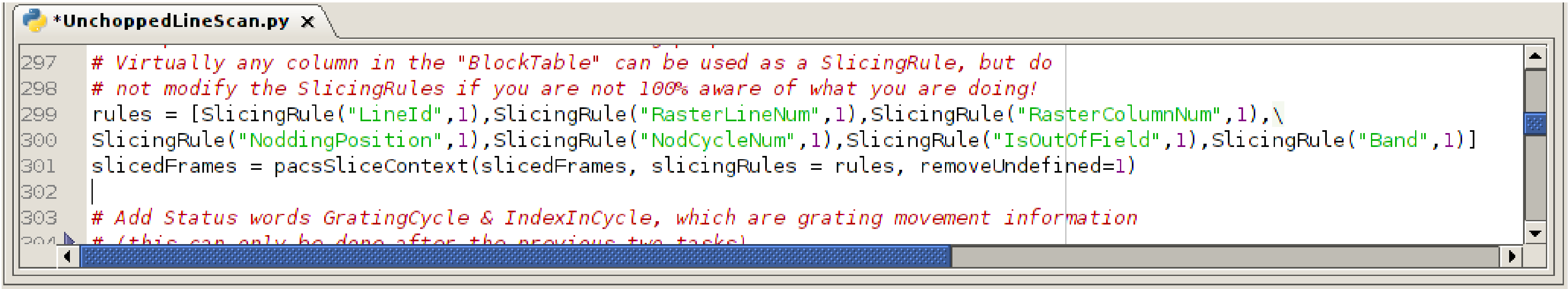


```
# Virtually any column in the "BlockTable" can be used as a SlicingRule, but do
# not modify the SlicingRules if you are not 100% aware of what you are doing!
rules = [SlicingRule("LineId",1),SlicingRule("RasterLineNum",1),SlicingRule("RasterColumnNum",1),\
SlicingRule("NoddingPosition",1),SlicingRule("NodCycleNum",1),SlicingRule("IsOutOfField",1),SlicingRule("Band",1)]
slicedFrames = pacsSliceContext(slicedFrames, slicingRules = rules, removeUndefined=1)

# Add Status words GratingCycle & IndexInCycle, which are grating movement information
# (this can only be done after the previous two tasks)
```

The slicing of the data is performed according to rules made explicit in the pipeline. In our example, one line is observed in four positions (ON, OFF, OFF, and ON). So, we expect 4 slices plus an initial slice containing the calibration block.

Slicing

NHSC PACS Web Tutorial



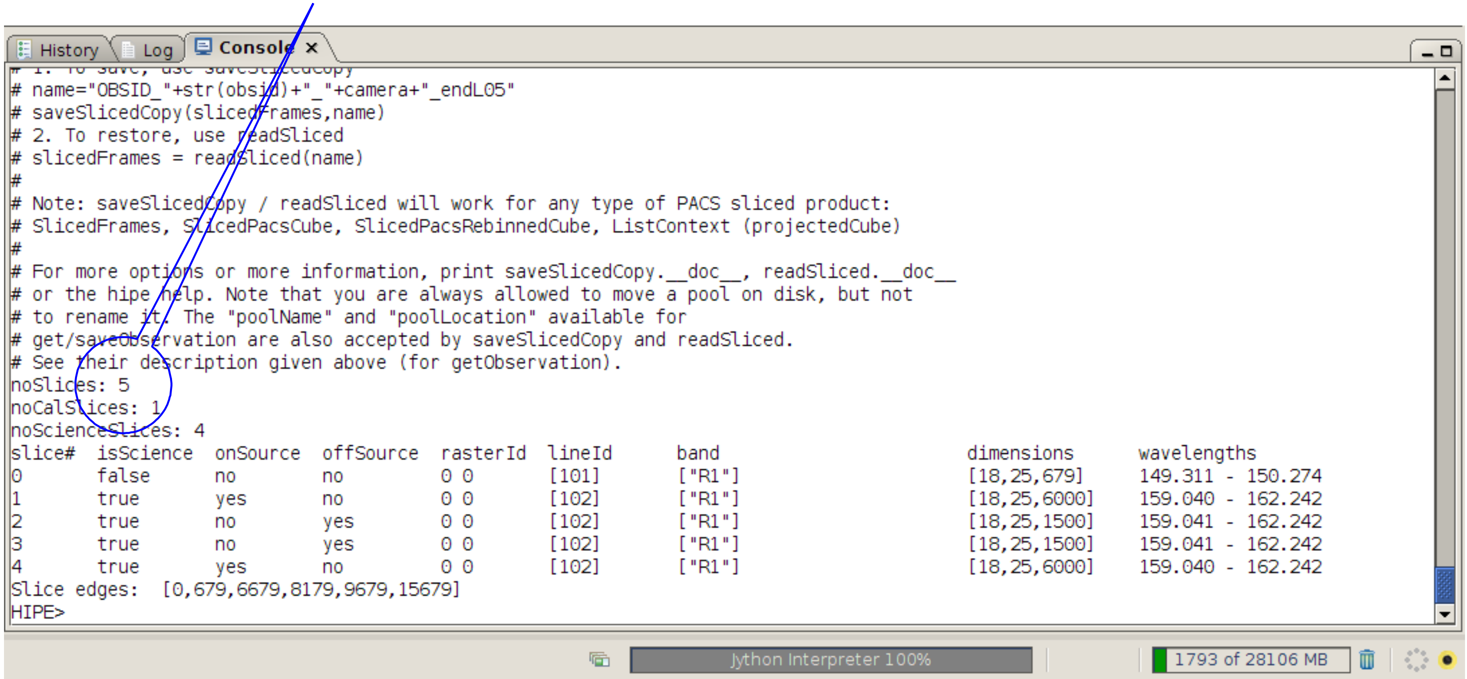
*UnchoppedLineScan.py

```
# Virtually any column in the "BlockTable" can be used as a SlicingRule, but do
# not modify the SlicingRules if you are not 100% aware of what you are doing!
rules = [SlicingRule("LineId",1),SlicingRule("RasterLineNum",1),SlicingRule("RasterColumnNum",1),\
SlicingRule("NoddingPosition",1),SlicingRule("NodCycleNum",1),SlicingRule("IsOutOfField",1),SlicingRule("Band",1)]
slicedFrames = pacsSliceContext(slicedFrames, slicingRules = rules, removeUndefined=1)

# Add Status words GratingCycle & IndexInCycle, which are grating movement information
# (this can only be done after the previous two tasks)
```

The slicing of the data is performed according to rules made explicit in the pipeline. In our example, one line is observed in four positions (ON, OFF, OFF, and ON). So, we expect 4 slices plus an initial slice containing the calibration block.

Check: after slicing

5 slices !



```
# 1. To save, use saveSlicedCopy
# name="OBSID_"+str(obsid)+"_"+camera+"_endL05"
# saveSlicedCopy(slicedFrames,name)
# 2. To restore, use readSliced
# slicedFrames = readSliced(name)
#
# Note: saveSlicedCopy / readSliced will work for any type of PACS sliced product:
# SlicedFrames, SlicedPacsCube, SlicedPacsRebinnedCube, ListContext (projectedCube)
#
# For more options or more information, print saveSlicedCopy.__doc__, readSliced.__doc__
# or the hipe help. Note that you are always allowed to move a pool on disk, but not
# to rename it. The "poolName" and "poolLocation" available for
# get/saveObservation are also accepted by saveSlicedCopy and readSliced.
# See their description given above (for getObservation).
noSlices: 5
noCalSlices: 1
noScienceSlices: 4
slice#  isScience  onSource  offSource  rasterId  lineId      band                   dimensions      wavelengths
0       false      no        no         0 0       [101]       ["R1"]                 [18,25,679]     149.311 - 150.274
1       true       yes       no         0 0       [102]       ["R1"]                 [18,25,6000]    159.040 - 162.242
2       true       no        yes        0 0       [102]       ["R1"]                 [18,25,1500]    159.041 - 162.242
3       true       no        yes        0 0       [102]       ["R1"]                 [18,25,1500]    159.041 - 162.242
4       true       yes       no         0 0       [102]       ["R1"]                 [18,25,6000]    159.040 - 162.242
Slice edges:  [0,679,6679,8179,9679,15679]
HIPE>
```

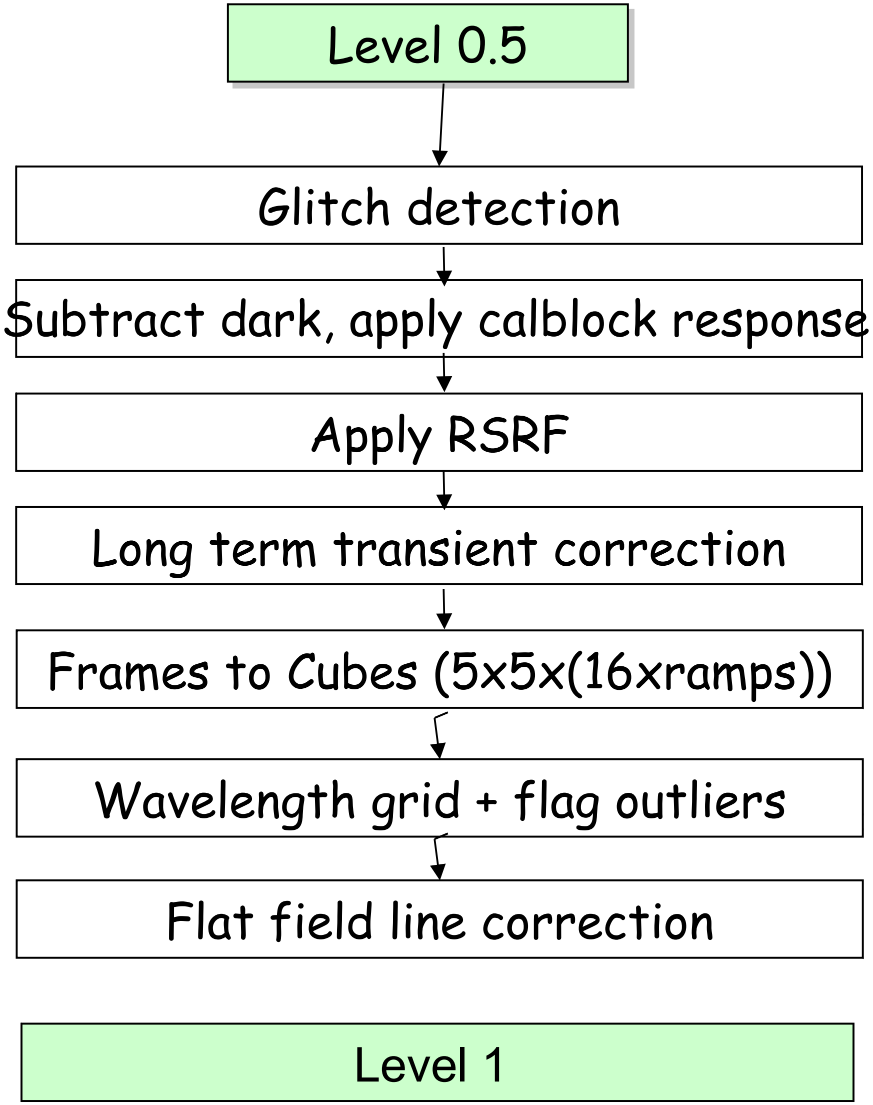
In the description we know which slice is on and off source, the wavelength range covered and the band. From this table, note the lineId number. This will be used later in the pipeline. In this case, we will display line 102.

Step 4

Run the 0.5 → 1 pipeline

Glitch detection, chop differentiation, RSRF, flat

Level 0.5 → 1



Level 0.5

Glitch detection

Subtract dark, apply calblock response

Apply RSRF

Long term transient correction

Frames to Cubes (5x5x(16xramps))

Wavelength grid + flag outliers

Flat field line correction

Level 1

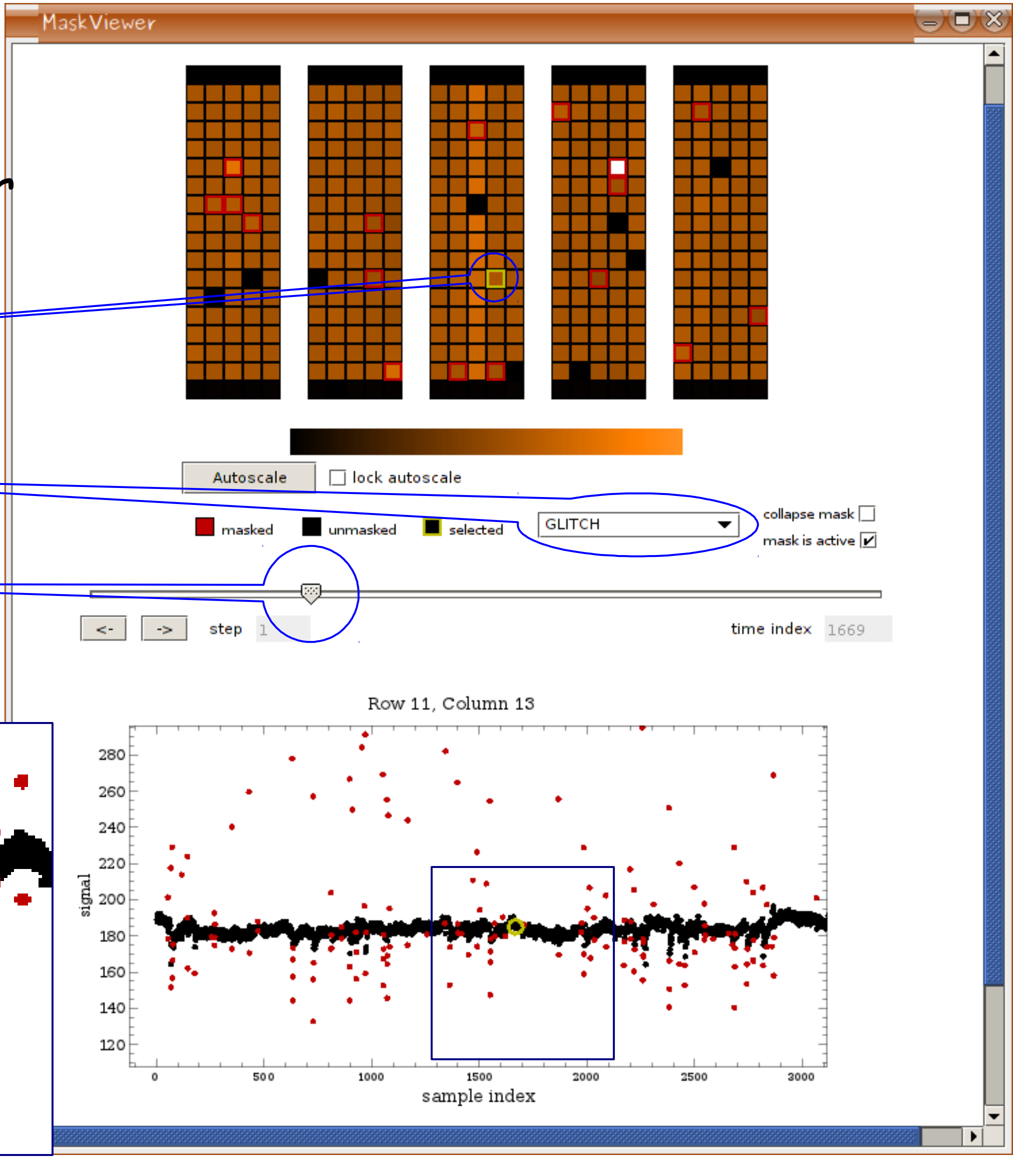
Glitch detection

You can check manually the points flagged as glitch or masked for other reasons using the maskviewer

Select a pixel by clicking on it

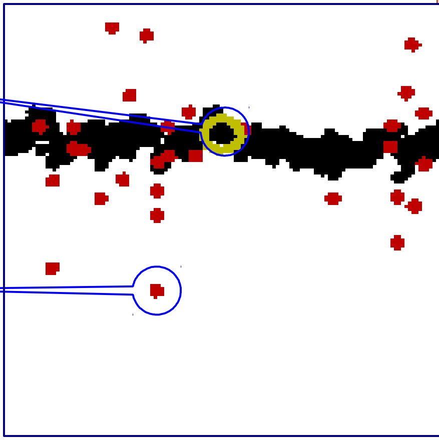
Select a mask

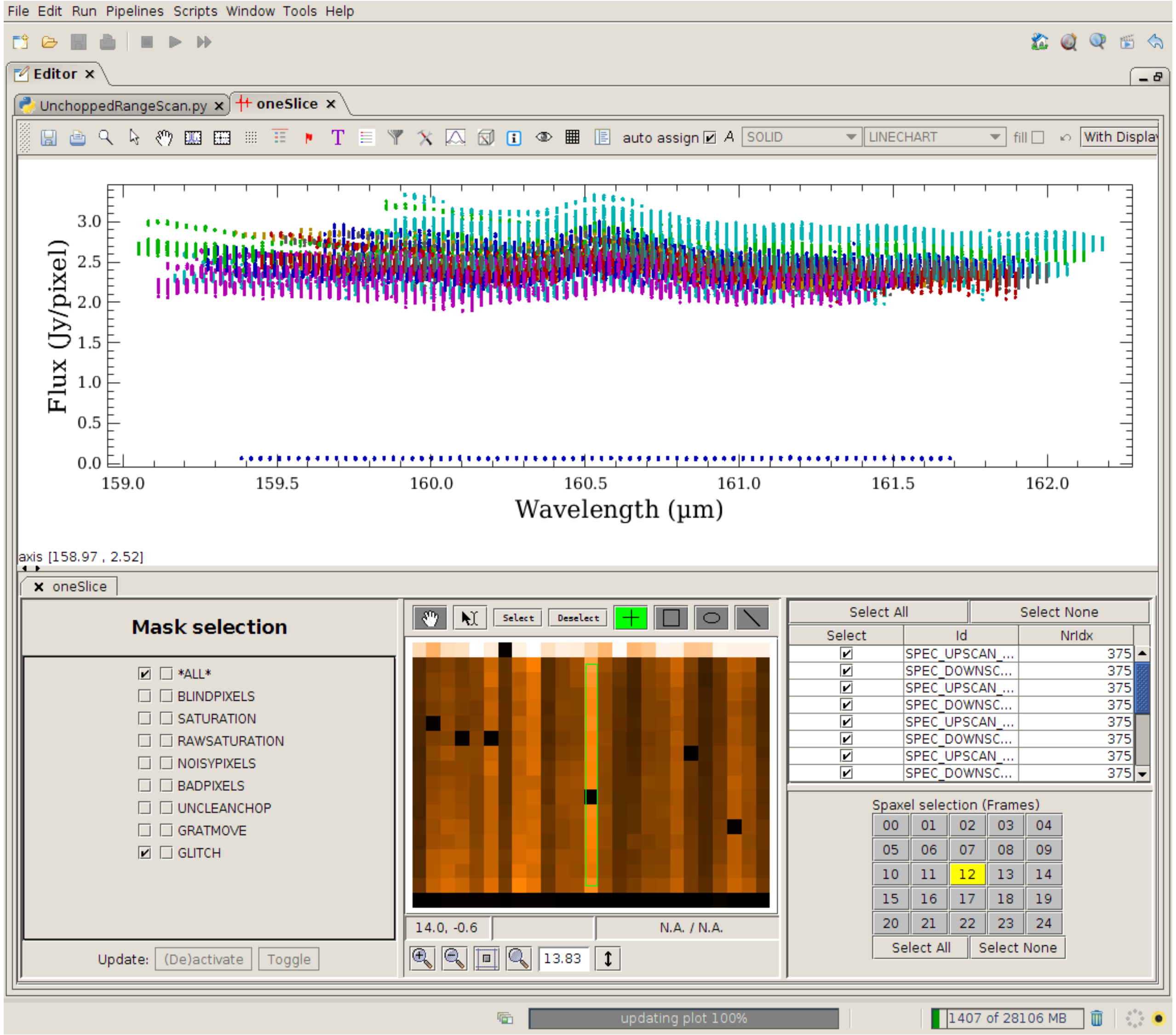
Select a frame



Current frame

Masked glitch





Mask selection

☑ □ *ALL*
□ □ BLINDPIXELS
□ □ SATURATION
□ □ RAWSATURATION
□ □ NOISYPIXELS
□ □ BADPIXELS
□ □ UNCLEANCHOP
□ □ GRATMOVE
☑ □ GLITCH

Select All | Select None

Select	Id	NrIdx
☑	SPEC_UPSCAN_...	375
☑	SPEC_DOWNSC...	375
☑	SPEC_UPSCAN_...	375
☑	SPEC_DOWNSC...	375
☑	SPEC_UPSCAN_...	375
☑	SPEC_DOWNSC...	375
☑	SPEC_UPSCAN_...	375
☑	SPEC_DOWNSC...	375

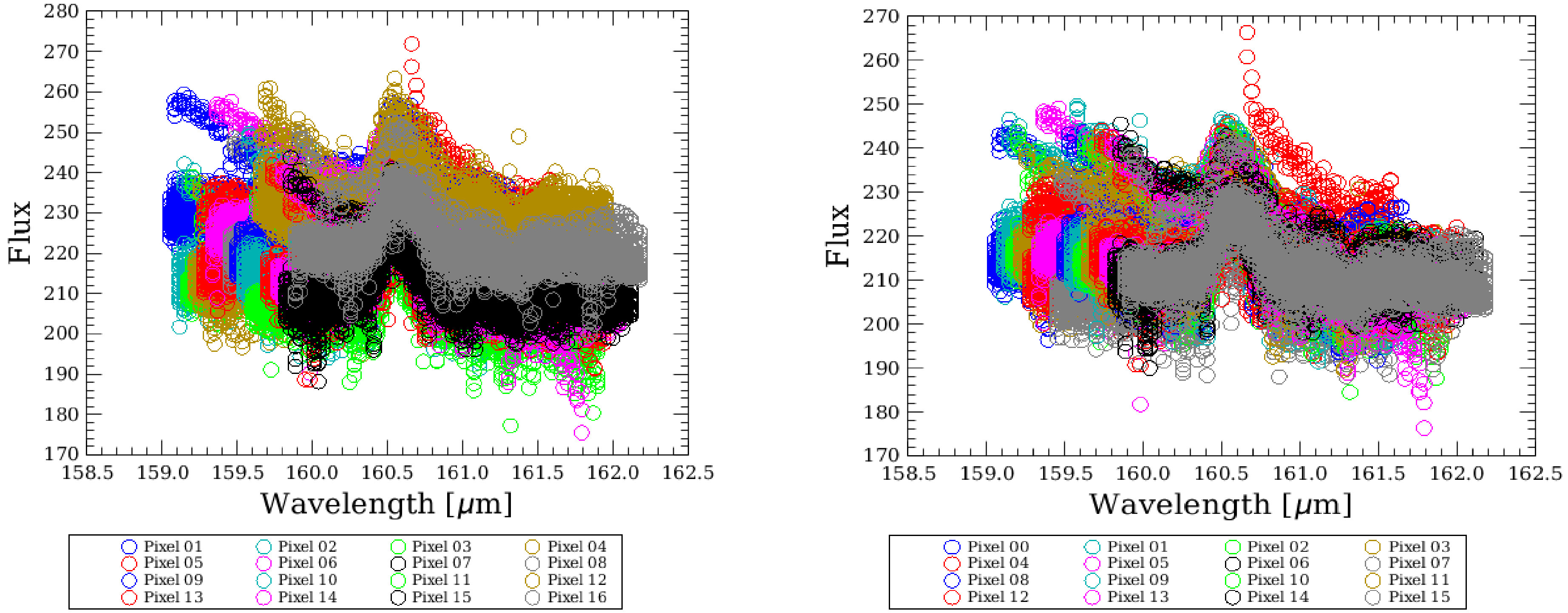
Spaxel selection (Frames)

00	01	02	03	04
05	06	07	08	09
10	11	12	13	14
15	16	17	18	19
20	21	22	23	24

Select All | Select None

Pixels can be now examined with the Spectrum Explorer

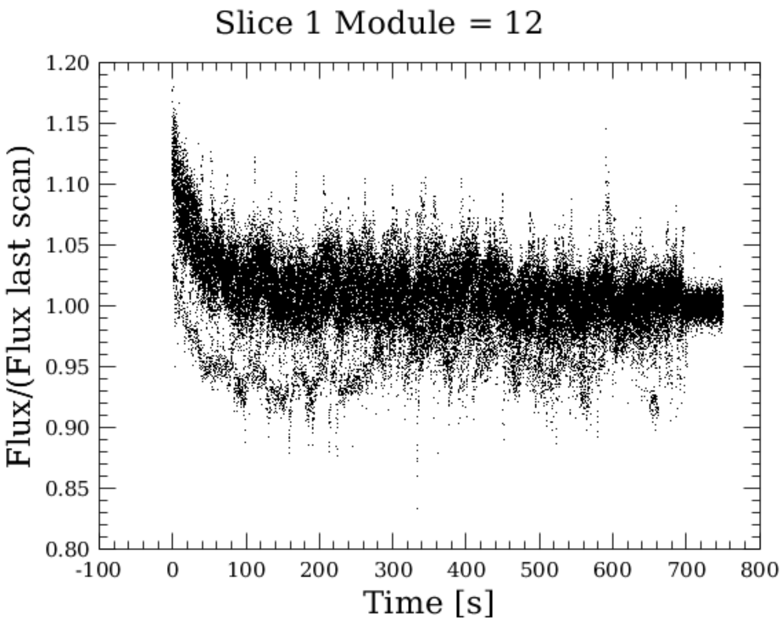
RSRF, Dark, Response

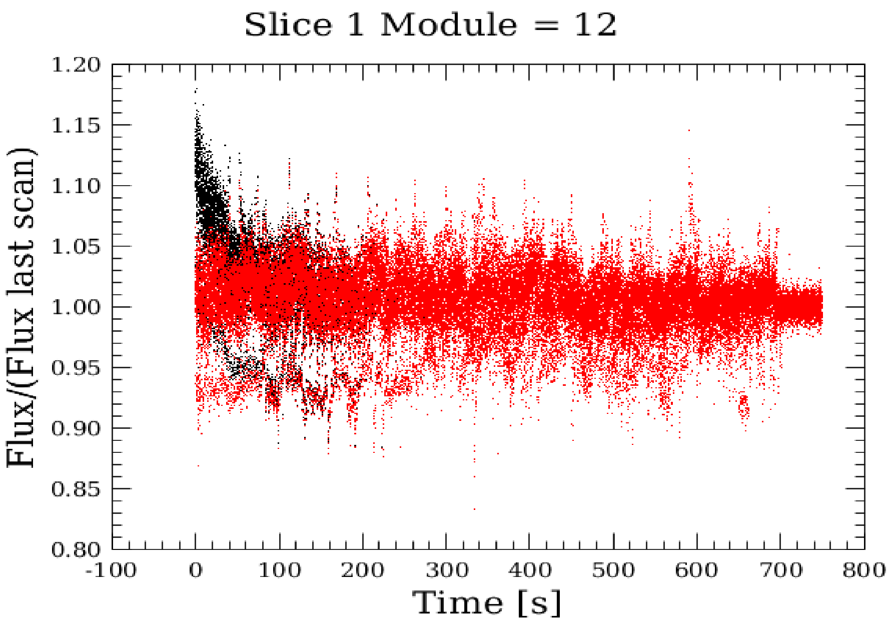


Central module before and after applying several corrections (dark, response, and RSRF)

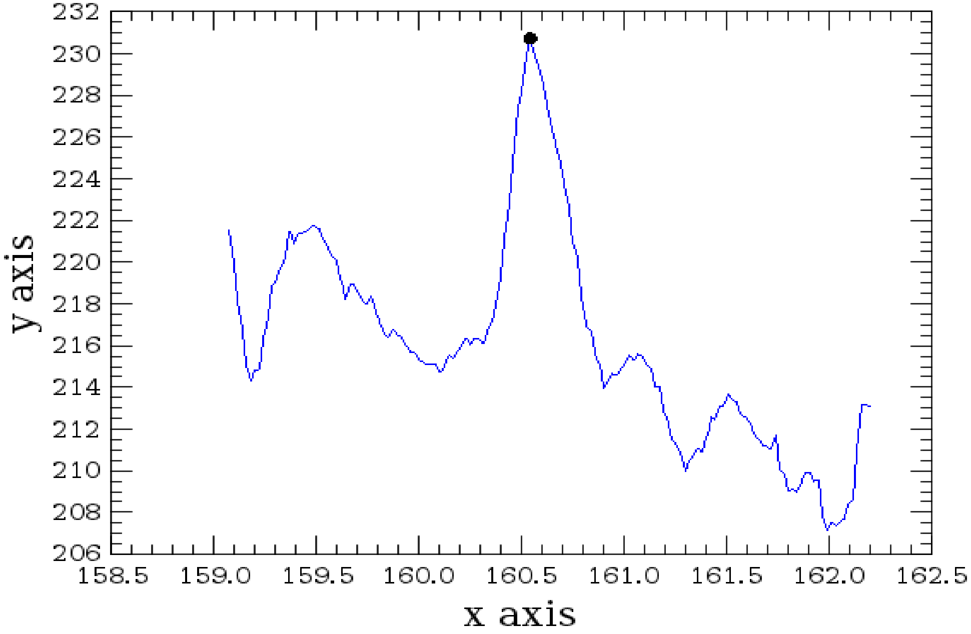
Long term transients

The effect of the transient correction is shown on the central module in the first slice after the calibration block. The plot shows the signal of the spectral pixels in the central module normalized to the signal in the last grating scan. Black and red are before and after the correction, respectively.



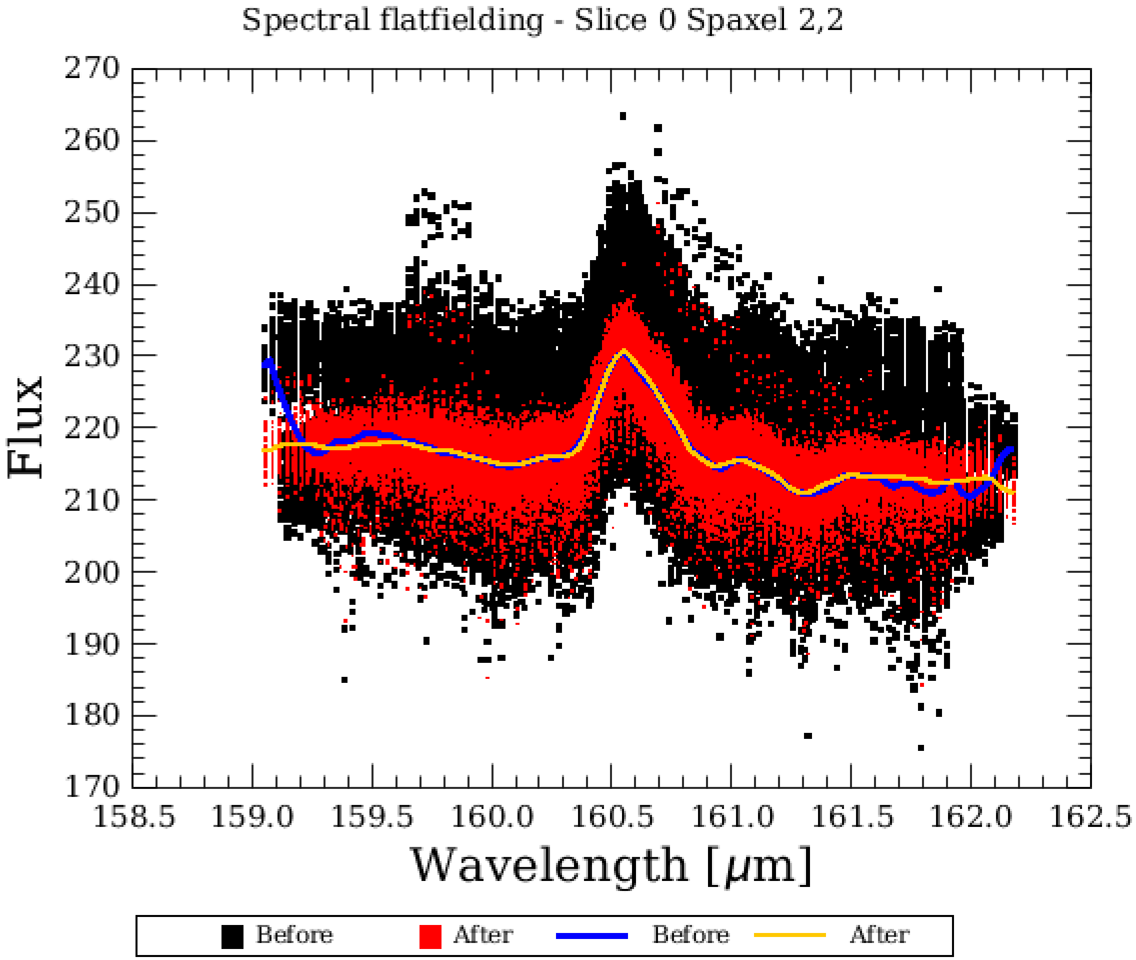


Spectral flat field

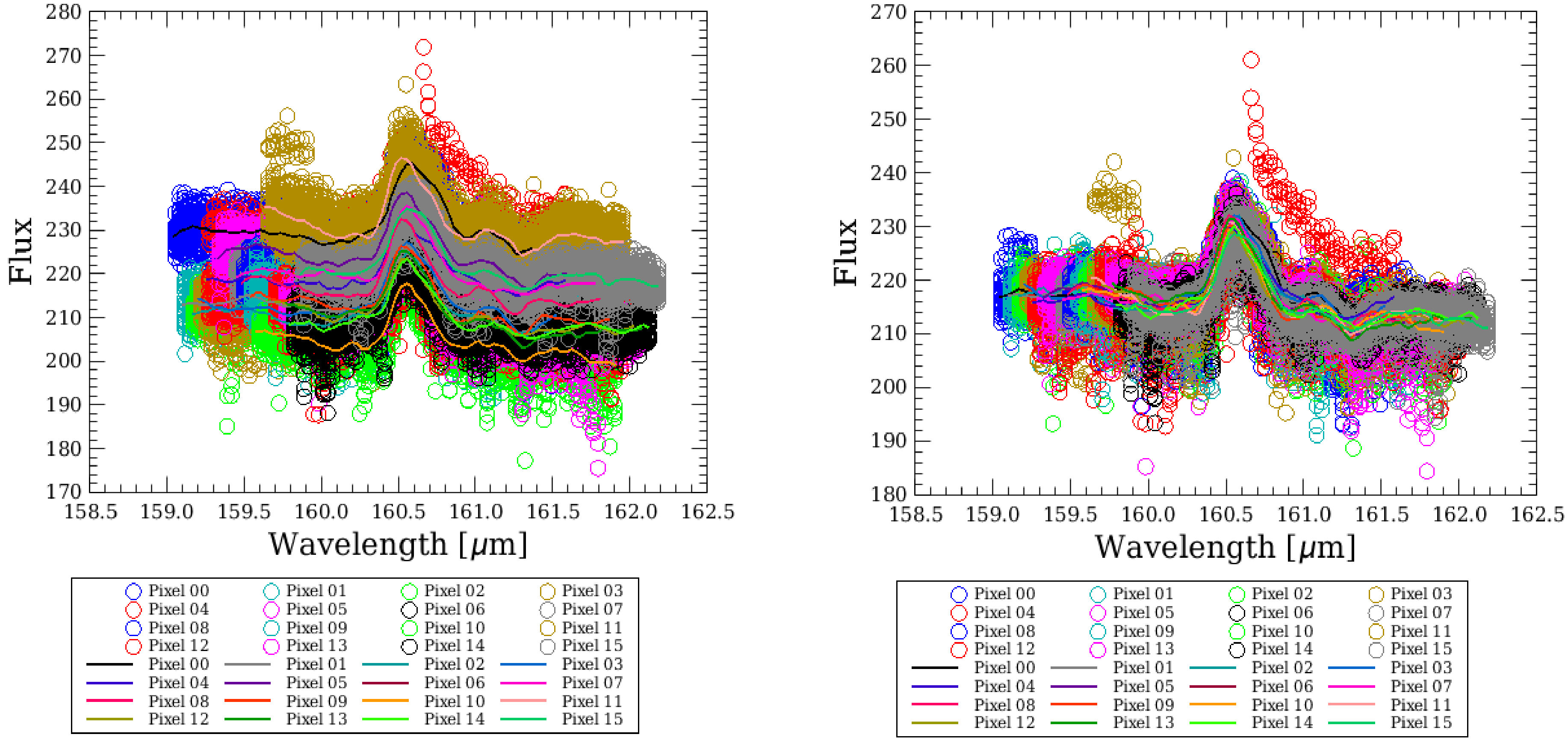


The spectral flat-field has been greatly improved in HIPE 8. Now, each module is explored to detect lines to avoid them when comparing the spectra from different spectral pixels. In verbose mode, the different spectra pop out and the line are identified with a black dot.

Spectral flat field

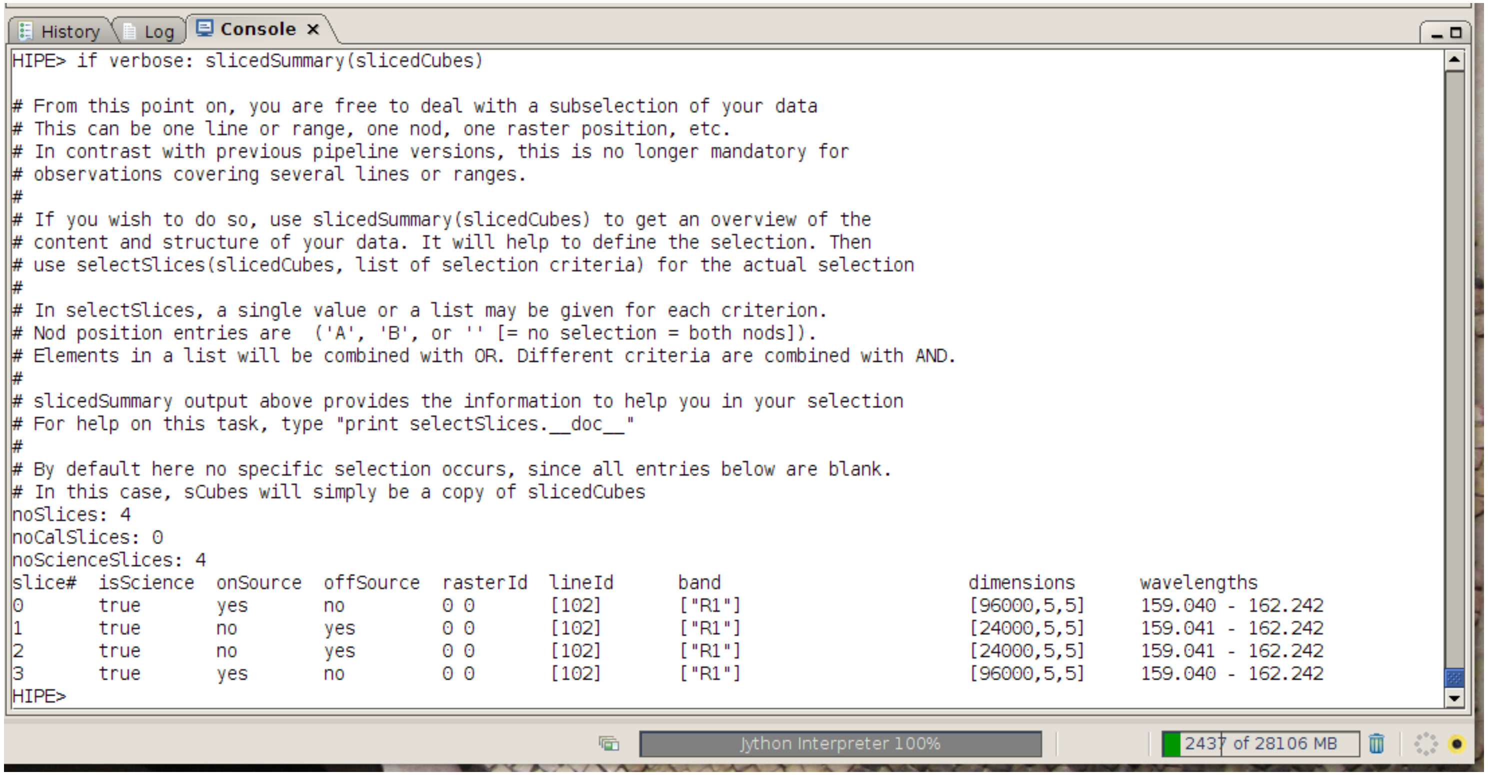


Spectral flat field



Central module before and after the first spectral flat-fielding
The transient correction has greatly improved the final result.

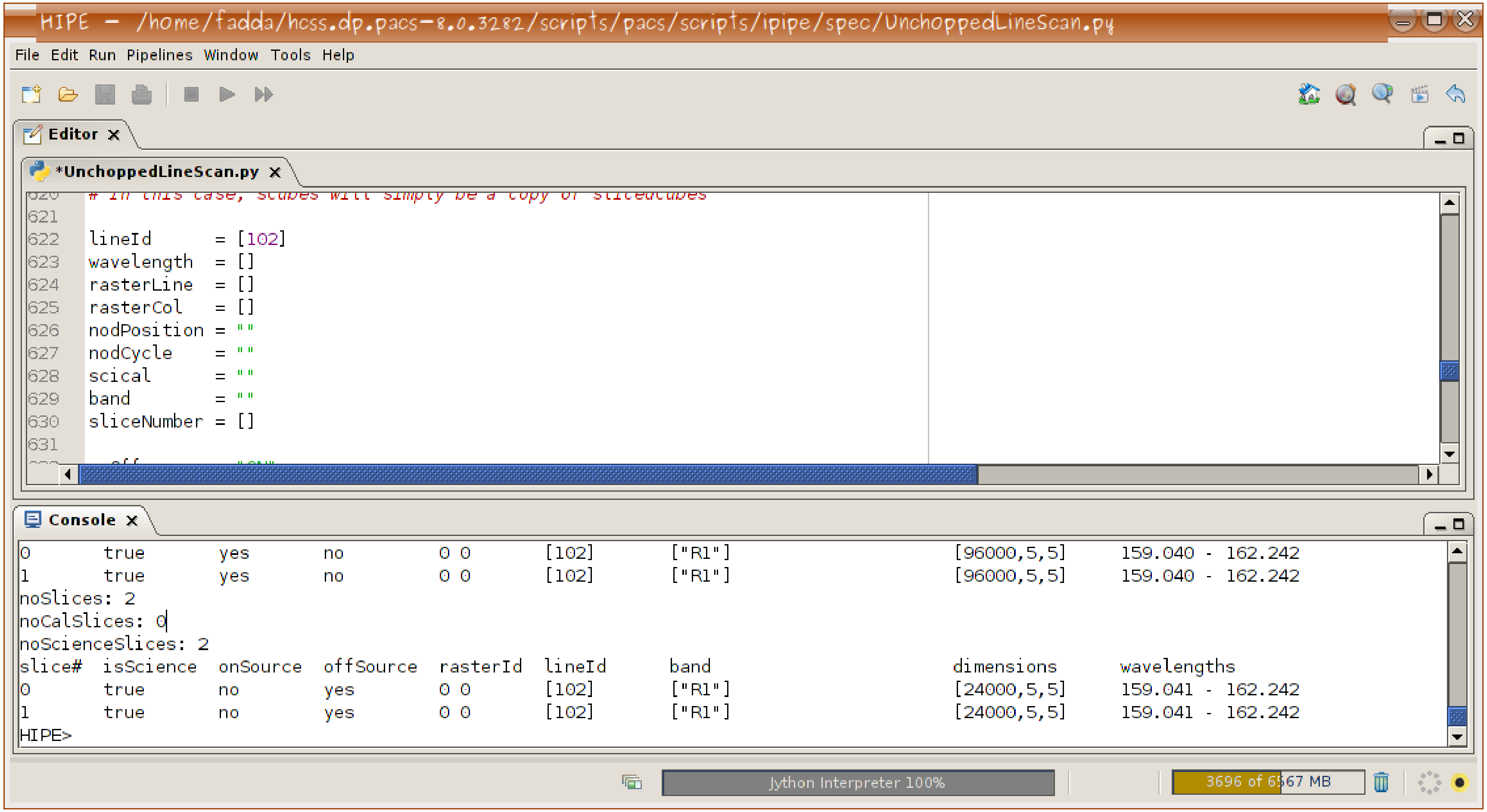
Level 1



```
HIPE> if verbose: slicedSummary(slicedCubes)

# From this point on, you are free to deal with a subselection of your data
# This can be one line or range, one nod, one raster position, etc.
# In contrast with previous pipeline versions, this is no longer mandatory for
# observations covering several lines or ranges.
#
# If you wish to do so, use slicedSummary(slicedCubes) to get an overview of the
# content and structure of your data. It will help to define the selection. Then
# use selectSlices(slicedCubes, list of selection criteria) for the actual selection
#
# In selectSlices, a single value or a list may be given for each criterion.
# Nod position entries are  ('A', 'B', or '' [= no selection = both nods]).
# Elements in a list will be combined with OR. Different criteria are combined with AND.
#
# slicedSummary output above provides the information to help you in your selection
# For help on this task, type "print selectSlices.__doc__"
#
# By default here no specific selection occurs, since all entries below are blank.
# In this case, sCubes will simply be a copy of slicedCubes
noSlices: 4
noCalSlices: 0
noScienceSlices: 4
slice#  isScience  onSource  offSource  rasterId  lineId      band                 dimensions     wavelengths
0       true       yes       no         0 0       [102]       ["R1"]               [96000,5,5]    159.040 - 162.242
1       true       no        yes        0 0       [102]       ["R1"]               [24000,5,5]    159.041 - 162.242
2       true       no        yes        0 0       [102]       ["R1"]               [24000,5,5]    159.041 - 162.242
3       true       yes       no         0 0       [102]       ["R1"]               [96000,5,5]    159.040 - 162.242
HIPE>
```

We have reached level 1. slicedSummary gives a list of the content of the cubes. Let's choose a line to go to level 2. Also have a look to tutorial PACS 302.

Level 2



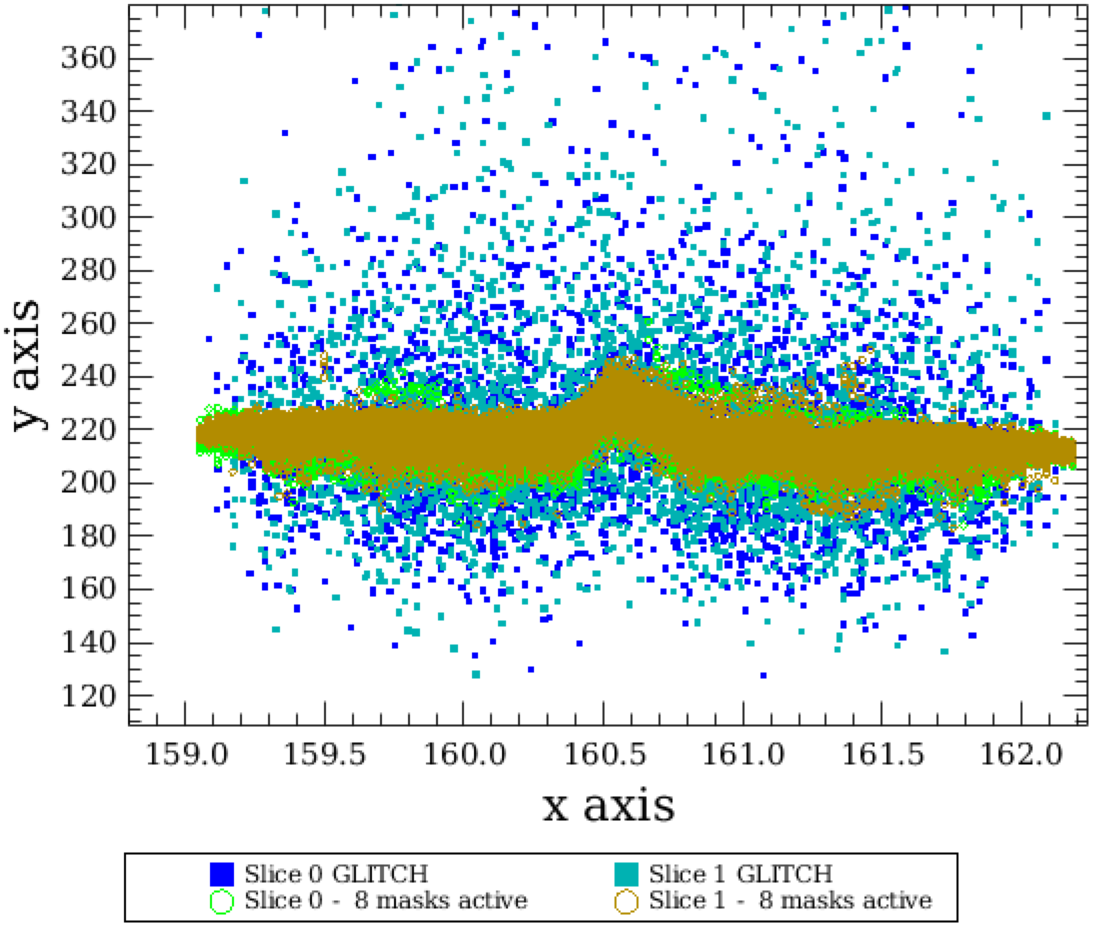
HIPE – /home/fadda/hcss.dp.pacs-8.0.3282/scripts/pacs/scripts/ipipe/spec/UnchoppedLineScan.py

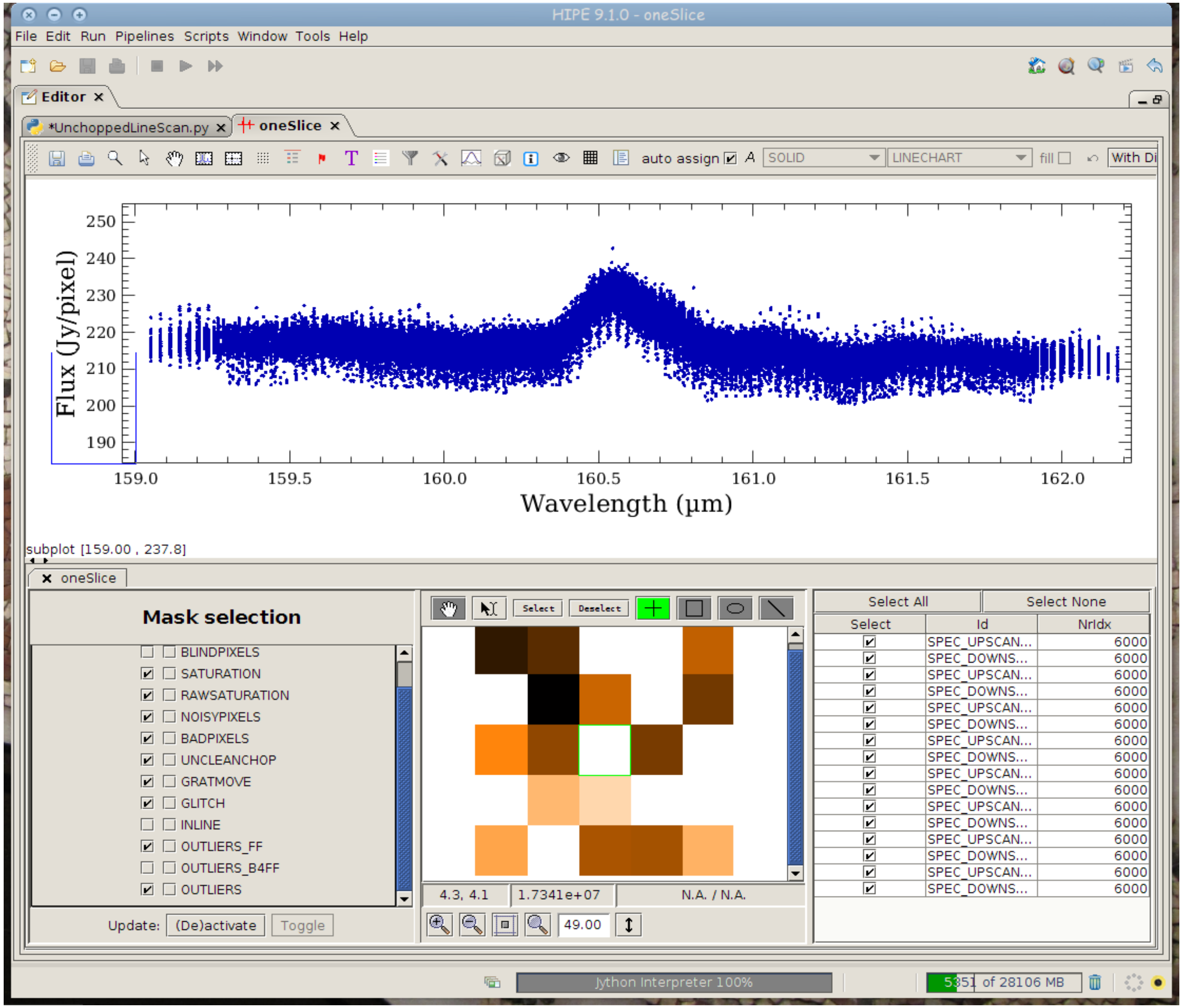
```
# In this case, scubes will simply be a copy of slicedcubes

lineId      = [102]
wavelength  = []
rasterLine  = []
rasterCol   = []
nodPosition = ""
nodCycle    = ""
scical      = ""
band        = ""
sliceNumber = []
```

```
0       true       yes       no        0 0      [102]       ["R1"]                   [96000,5,5]     159.040 - 162.242
1       true       yes       no        0 0      [102]       ["R1"]                   [96000,5,5]     159.040 - 162.242
noSlices: 2
noCalSlices: 0
noScienceSlices: 2
slice#  isScience  onSource  offSource  rasterId  lineId      band                     dimensions      wavelengths
0       true       no        yes       0 0      [102]       ["R1"]                   [24000,5,5]     159.041 - 162.242
1       true       no        yes       0 0      [102]       ["R1"]                   [24000,5,5]     159.041 - 162.242
HIPE>
```

In our case, we can only choose line 102.

Habemus spectrum !





subplot [159.00 , 237.8]

oneSlice

Mask selection

- □ □ BLINDPIXELS
- ☑ □ SATURATION
- ☑ □ RAWSATURATION
- ☑ □ NOISYPIXELS
- ☑ □ BADPIXELS
- ☑ □ UNCLEANCHOP
- ☑ □ GRATMOVE
- ☑ □ GLITCH
- □ □ INLINE
- ☑ □ OUTLIERS_FF
- □ □ OUTLIERS_B4FF
- ☑ □ OUTLIERS

4.3, 4.1 | 1.7341e+07 | N.A. / N.A.

49.00

Select	Id	NrIdx
☑	SPEC_UPSCAN...	6000
☑	SPEC_DOWNS...	6000
☑	SPEC_UPSCAN...	6000
☑	SPEC_DOWNS...	6000
☑	SPEC_UPSCAN...	6000
☑	SPEC_DOWNS...	6000
☑	SPEC_UPSCAN...	6000
☑	SPEC_DOWNS...	6000
☑	SPEC_UPSCAN...	6000
☑	SPEC_DOWNS...	6000
☑	SPEC_UPSCAN...	6000
☑	SPEC_DOWNS...	6000
☑	SPEC_UPSCAN...	6000
☑	SPEC_DOWNS...	6000
☑	SPEC_UPSCAN...	6000
☑	SPEC_DOWNS...	6000

5351 of 28106 MB

We can inspect the line with Spectrum Explorer

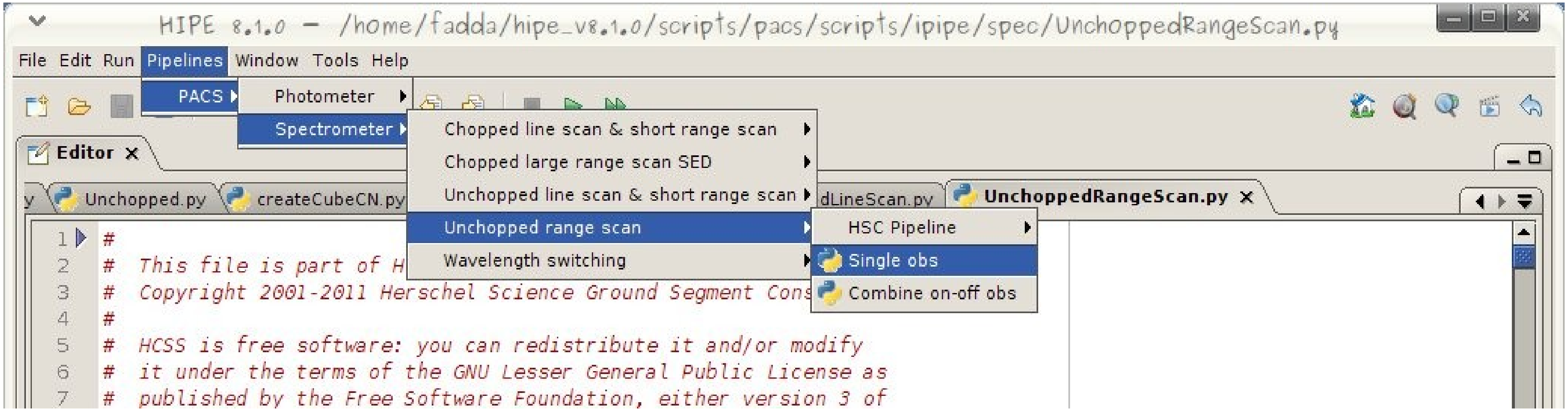
Unchopped range scan

Also for unchopped range scan it is possible to run an interactive script. The difference with the unchopped line scan is that:

A) there is no transient correction module

B) ON and OFF source observations are done in different observations. So two obs-ID numbers are required to reduce the observation properly.
This is done using two scripts: one to reduce each obs-ID and another one to combine them.

Unchopped range scan


HIPE 8.1.0 – /home/fadda/hipe_v8.1.0/scripts/pacs/scripts/ipipe/spec/UnchoppedRangeScan.py
File Edit Run Pipelines Window Tools Help
PACS
Photometer
Spectrometer
Chopped line scan & short range scan
Chopped large range scan SED
Unchopped line scan & short range scan
Unchopped range scan
Wavelength switching
HSC Pipeline
Single obs
Combine on-off obs
Editor
Unchopped.py
createCubeCN.py
dLineScan.py
UnchoppedRangeScan.py

```
#
#  This file is part of H
#  Copyright 2001-2011 Herschel Science Ground Segment Cons
#
#  HCSS is free software: you can redistribute it and/or modify
#  it under the terms of the GNU Lesser General Public License as
#  published by the Free Software Foundation, either version 3 of
```

The two interactive scripts available to reduce range unchopped observations:

a) Single obs for the ON and OFF observations

b) Combine the two AORs.