NHSC PACS
Web Tutorial

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 2013

Introduction

This tutorial will guide you through the interactive reduction of spectra obtained in unchopped line mode. This pipeline works for three different modes:

(i) unchopped line standard;
(ii) unchopped bright line;
(iii) wavelength-switching.

We remind that the archive reduction does not include the transient correction available in the interactive pipeline.

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

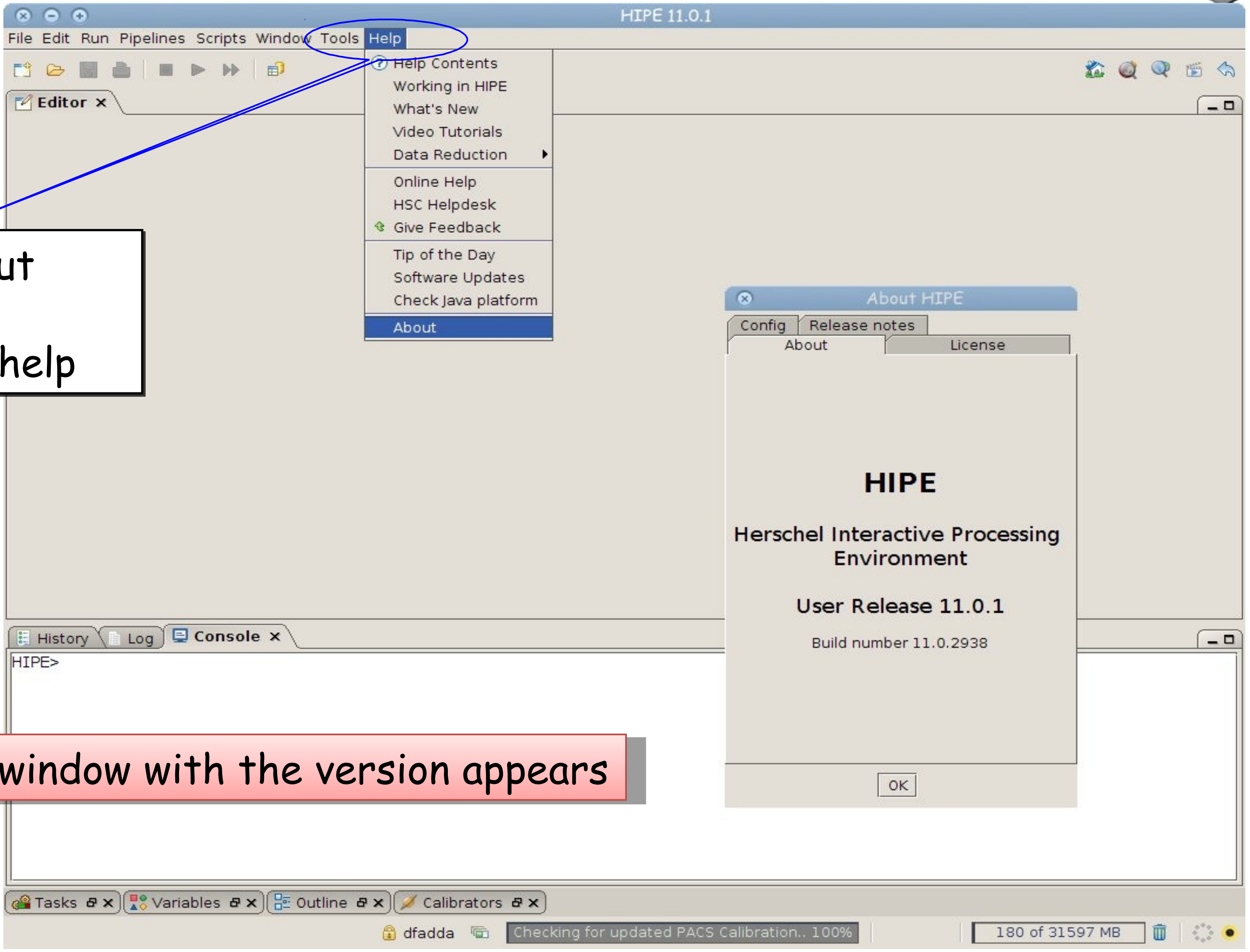
Step 3 Run the 0 → 0.5 pipeline

Step 4 Run the 0.5 → 1 pipeline

Step 1

Check HIPE version, memory allocation, and calibration products

The version used for the tutorial is 11.0.1



HIPE>

HIPE

Herschel Interactive Processing Environment

User Release 11.0.1

Build number 11.0.2938

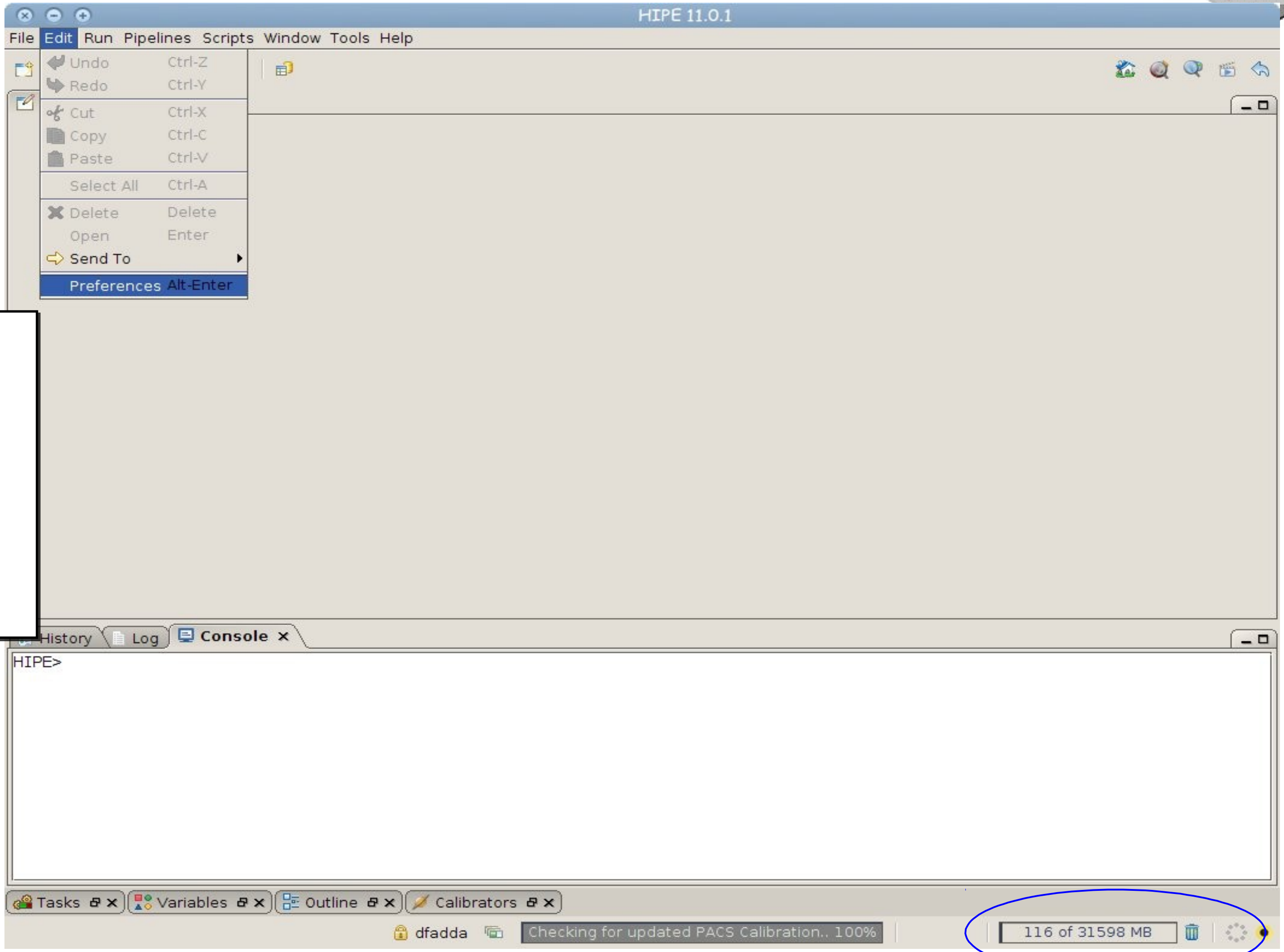
dfadda

Checking for updated PACS Calibration.. 100%

180 of 31597 MB

Select about from
drop down help

A pop-out window with the version appears



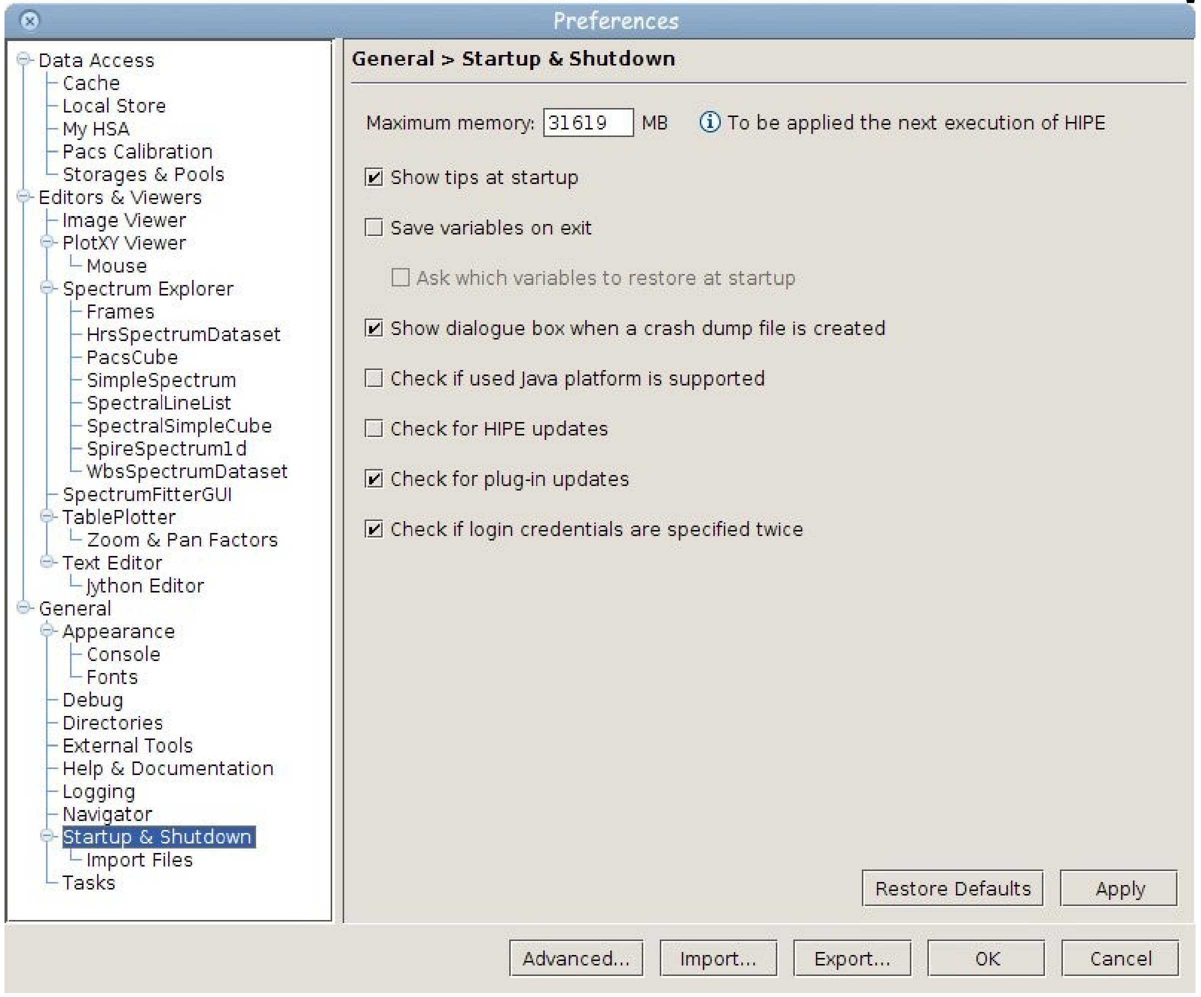
```
HIPE>
```

Checking for updated PACS Calibration.. 100%

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

Memory used and available

nhsc.ipac.caltech.edu/helpdesk



General > Startup & Shutdown

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

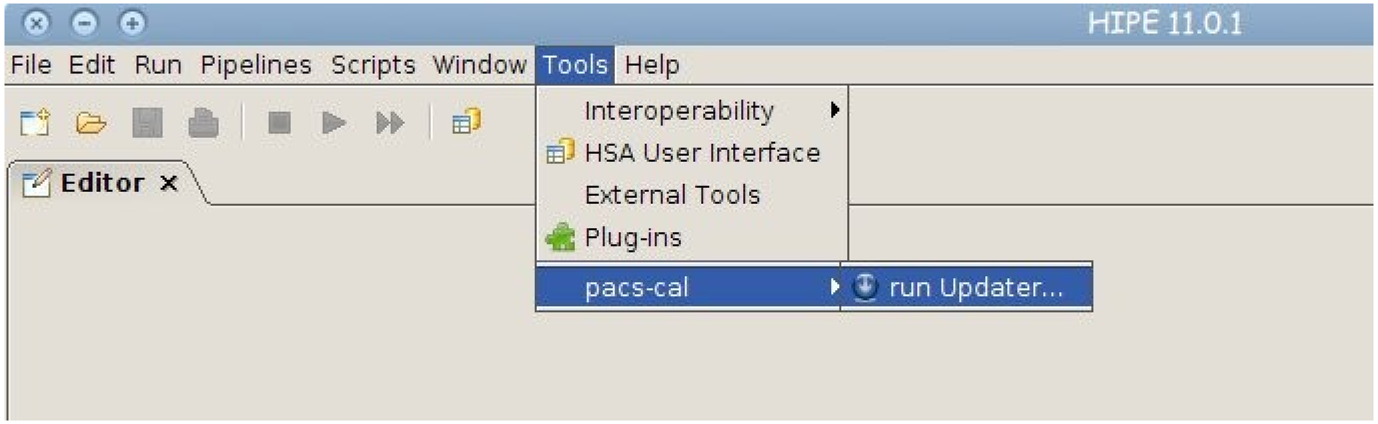
[✓] Show tips at startup
[ ] Save variables on exit
[ ] Ask which variables to restore at startup
[✓] Show dialogue box when a crash dump file is created
[ ] Check if used Java platform is supported
[ ] Check for HIPE updates
[✓] Check for plug-in updates
[✓] Check if login credentials are specified twice

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

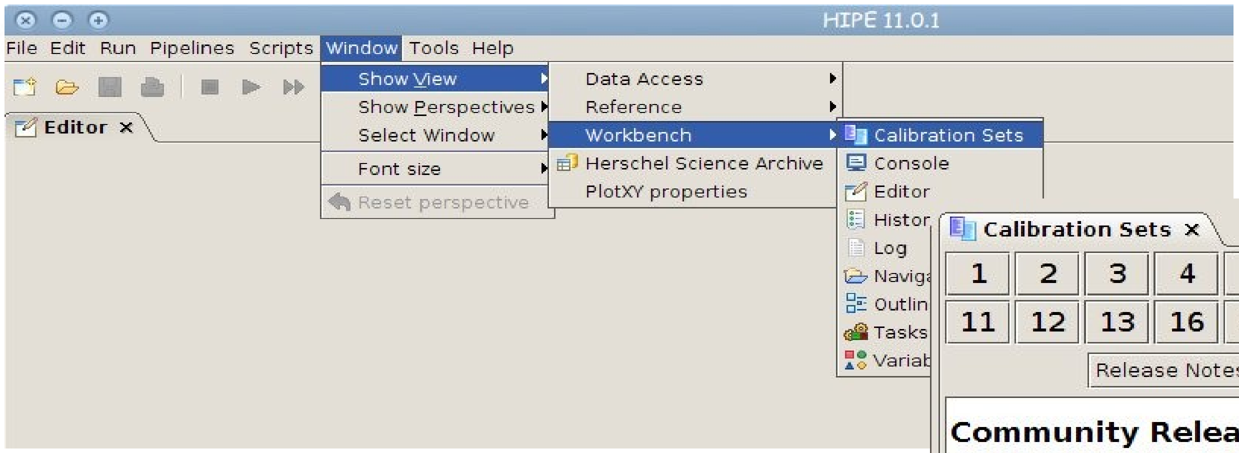
The allocated memory should be smaller than the total RAM of your computer. You have to exit and start a new session to use the new amount of memory.

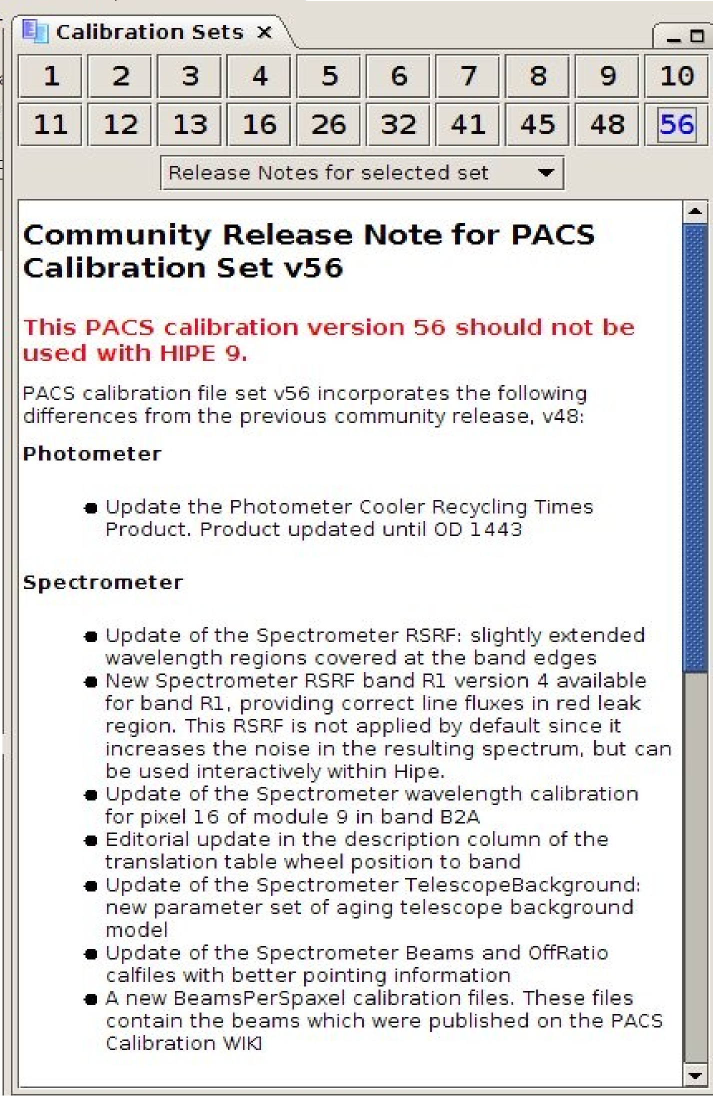
Calibration

Before running a new reduction, it is a safe habit to check if the latest calibration products are installed. The way to do it is running the Updater.



NHSC PACS Web Tutorial





Community Release Note for PACS Calibration Set v56

This PACS calibration version 56 should not be used with HIPE 9.

PACS calibration file set v56 incorporates the following differences from the previous community release, v48:

Photometer

- Update the Photometer Cooler Recycling Times Product. Product updated until OD 1443

Spectrometer

- Update of the Spectrometer RSRF: slightly extended wavelength regions covered at the band edges
- New Spectrometer RSRF band R1 version 4 available for band R1, providing correct line fluxes in red leak region. This RSRF is not applied by default since it increases the noise in the resulting spectrum, but can be used interactively within Hipe.
- Update of the Spectrometer wavelength calibration for pixel 16 of module 9 in band B2A
- Editorial update in the description column of the translation table wheel position to band
- Update of the Spectrometer TelescopeBackground: new parameter set of aging telescope background model
- Update of the Spectrometer Beams and OffRatio calfiles with better pointing information
- A new BeamsPerSpaxel calibration files. These files contain the beams which were published on the PACS Calibration WIKI

To inspect in detail what is different in the several releases of calibration products, it is possible to access the release notes for any release as well as the list of products with their specific versions.

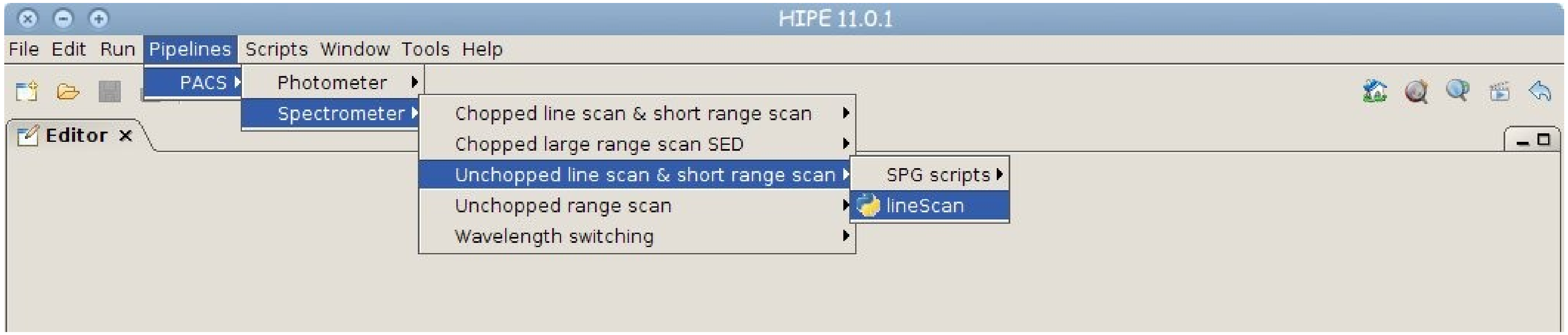
Step 2

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.

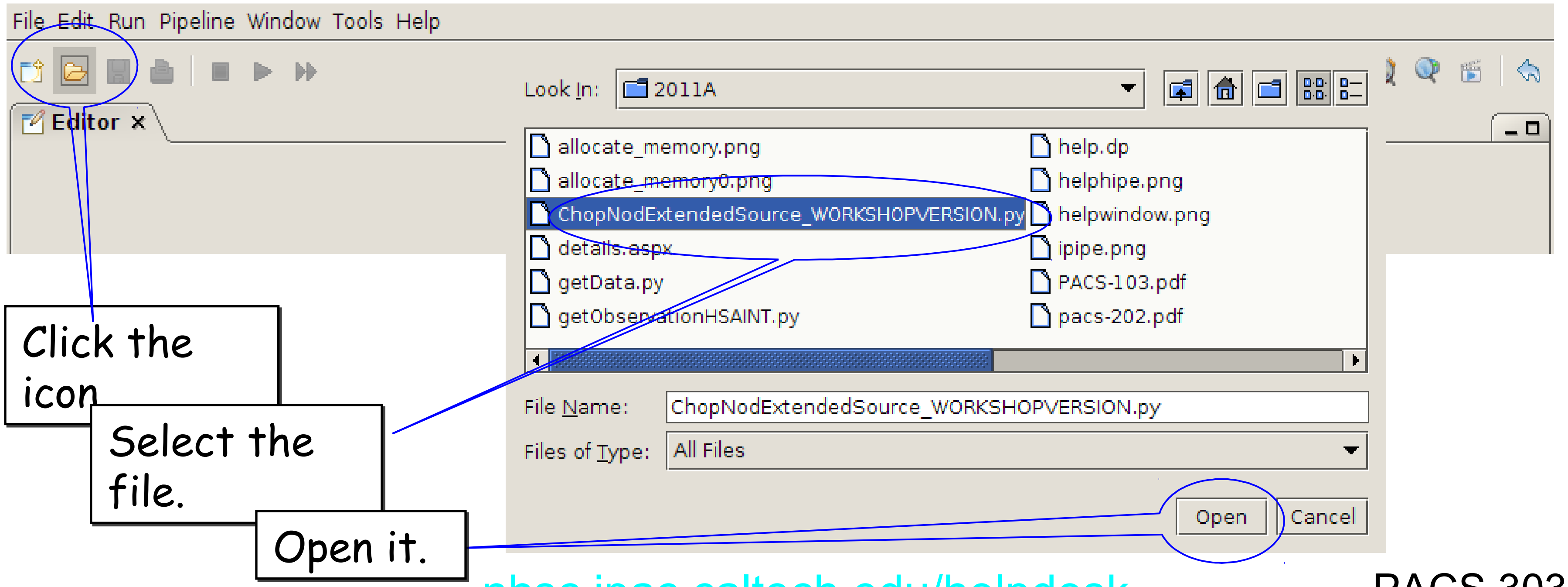

HIPE 11.0.1
File Edit Run Pipelines Scripts Window Tools Help
PACS
Photometer
Spectrometer
Chopped line scan & short range scan
Chopped large range scan SED
Unchopped line scan & short range scan
Unchopped range scan
Wavelength switching
SPG scripts
lineScan
Editor

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

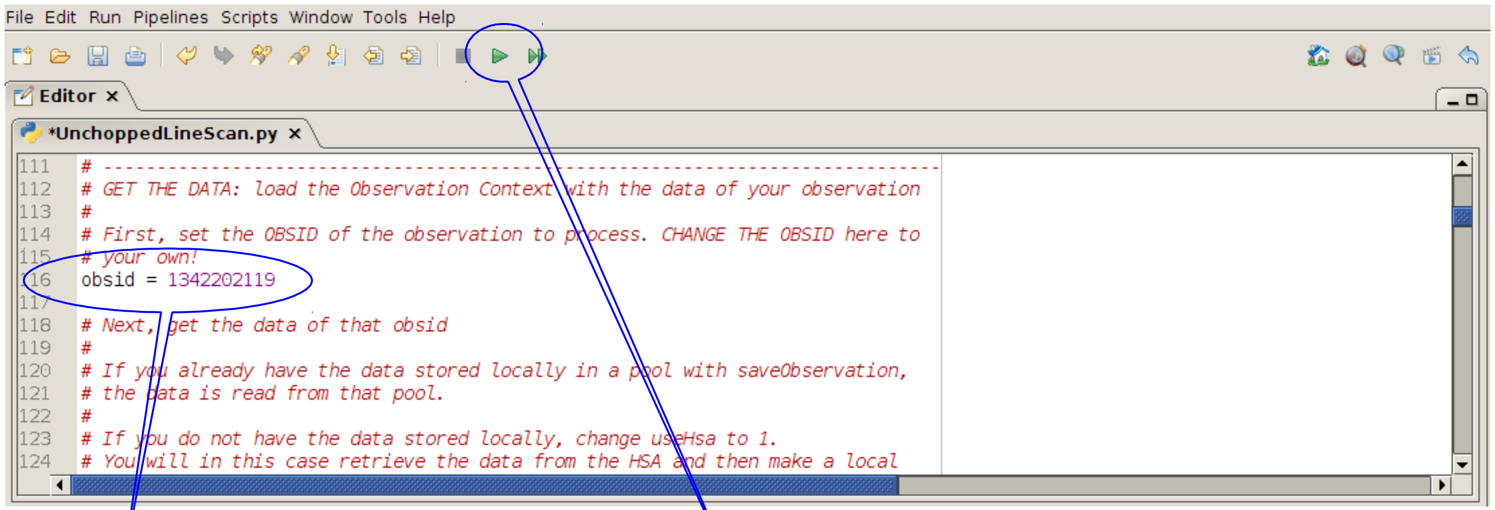
To load a custom script into an HIPE session, just click on the loading icon as shown in the figure. The search the location where you put the file using the pop-up window and finally load it into the session.



Loading the observation

NHSC PACS Web Tutorial

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.



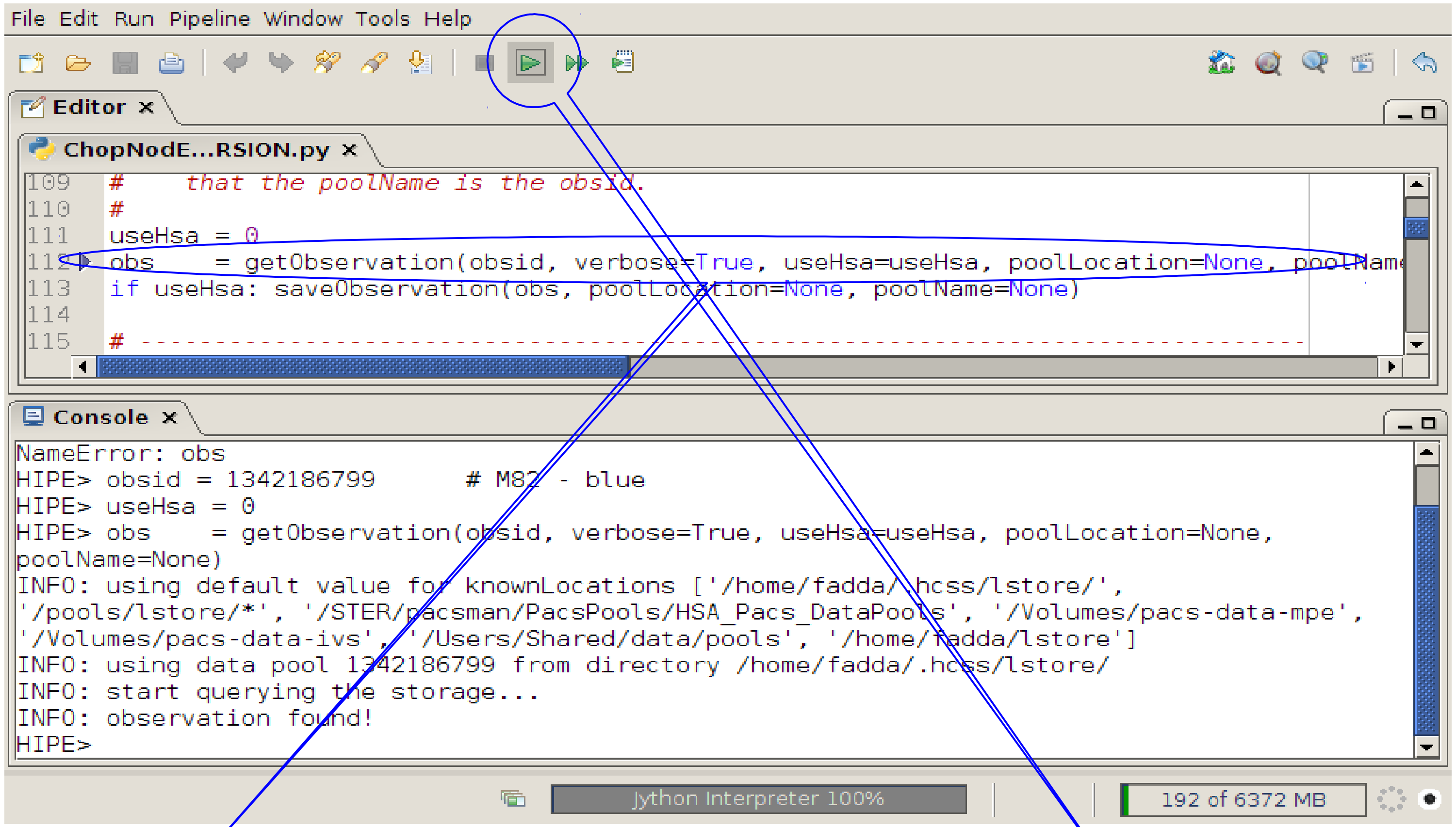
```
# ----------------------------------------------------------------------------
# GET THE DATA: load the Observation Context with the data of your observation
#
# First, set the OBSID of the observation to process. CHANGE THE OBSID here to
# your own!
obsid = 1342202119

# 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

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, poolName
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

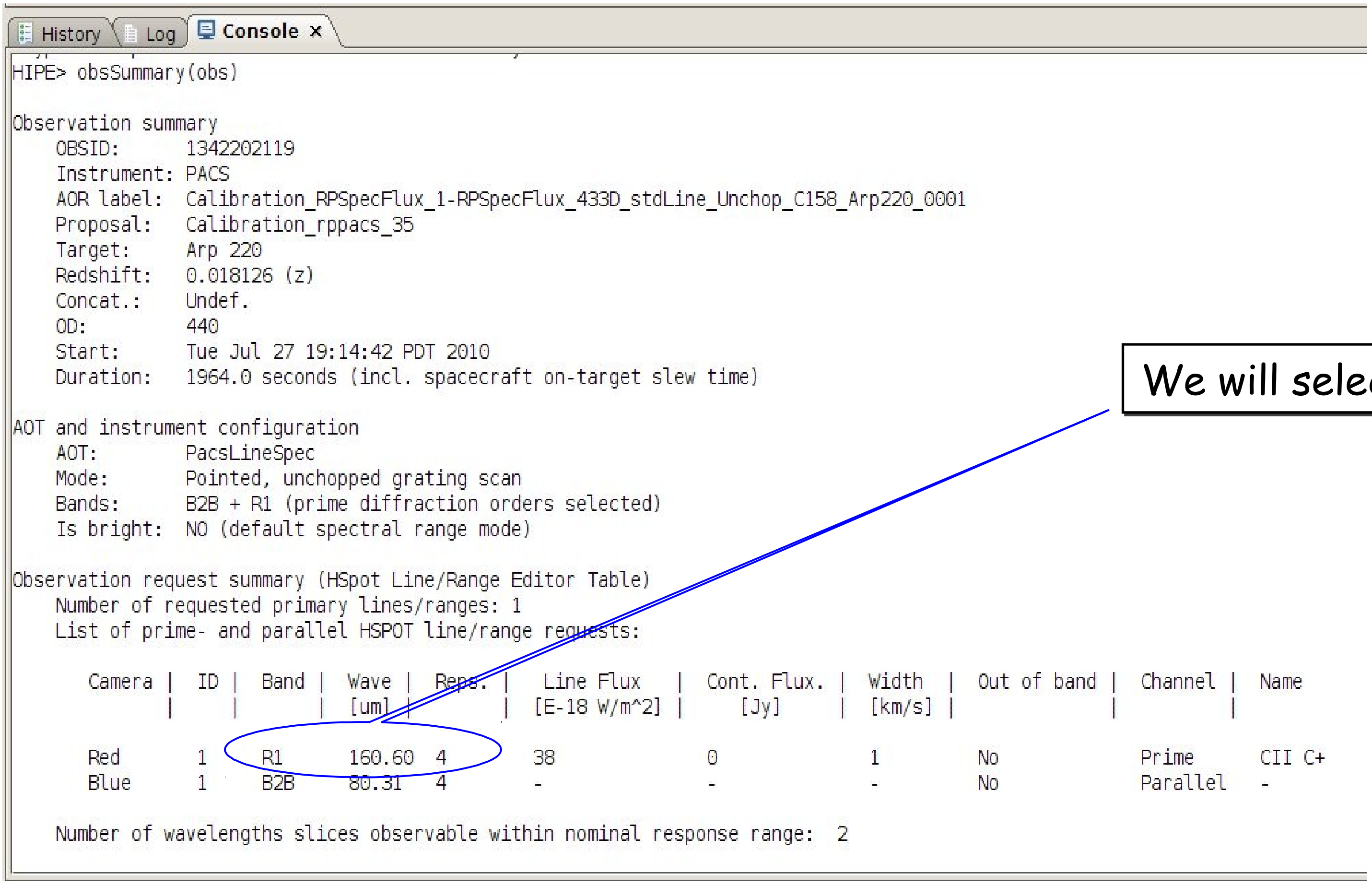
Check: observation summary

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



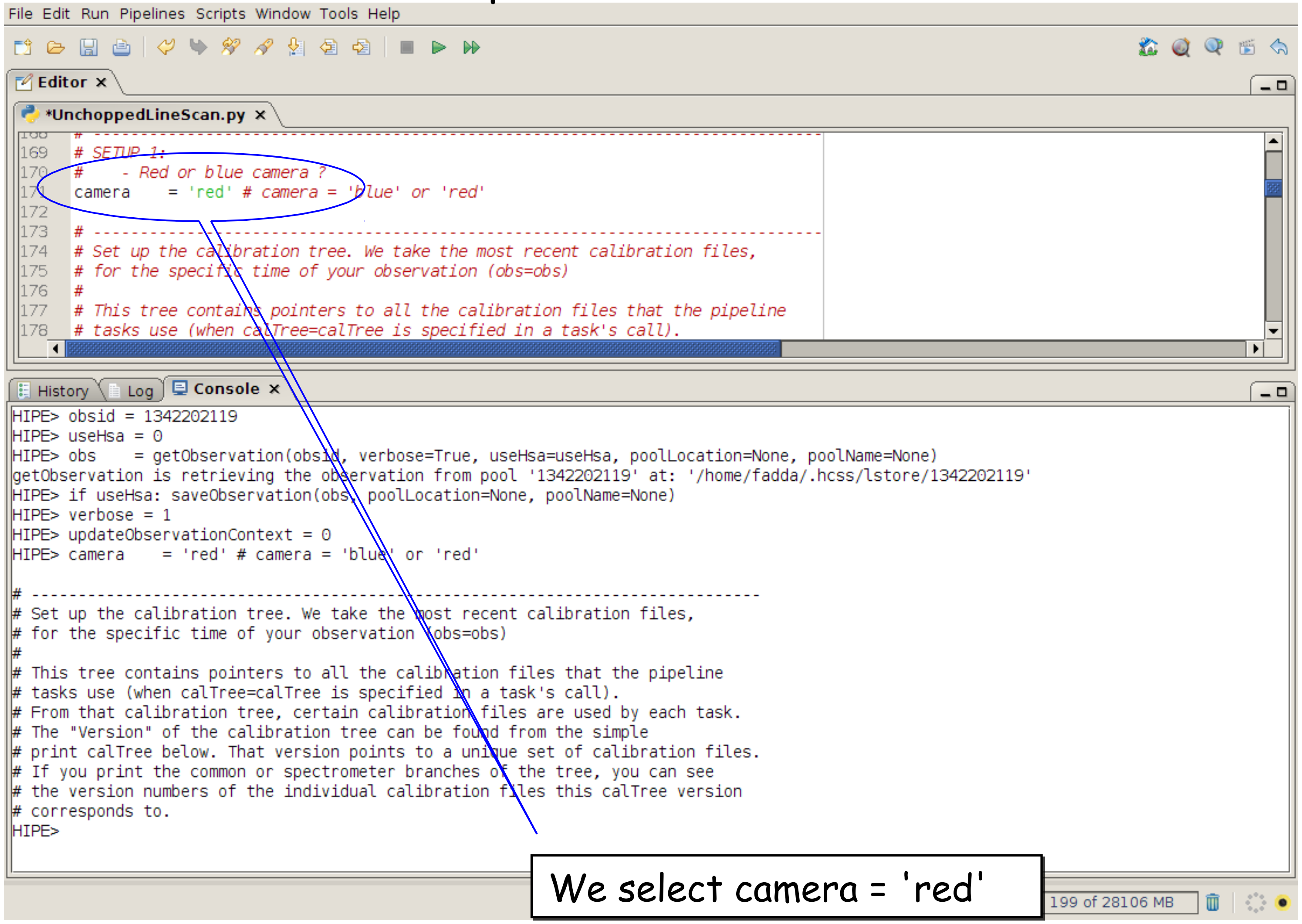
```
HIPE> obsSummary(obs)

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:  NO (default spectral range mode)

Observation request summary (HSpot Line/Range Editor Table)
    Number of requested primary lines/ranges: 1
    List of prime- and parallel HSPOT line/range requests:

       Camera |  ID |  Band |  Wave  | Reps. |  Line Flux    | Cont. Flux. |  Width  | Out of band |  Channel |  Name
              |     |       |  [um]  |       | [E-18 W/m^2]  |    [Jy]     | [km/s]  |             |          |

       Red      1     R1      160.60   4       38              0             1         No            Prime      CII C+
       Blue     1     B2B     80.31    4       -               -             -         No            Parallel   -

    Number of wavelengths slices observable within nominal response range:  2
```

We will select: camera = 'red'

nhsc.ipac.caltech.edu/helpdesk

Setting the camera

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



File Edit Run Pipelines Scripts Window Tools Help

Editor ×

*UnchoppedLineScan.py ×

```
# ---------------------------------------------------------------------
# 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).
```

History Log Console ×

```
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>
```

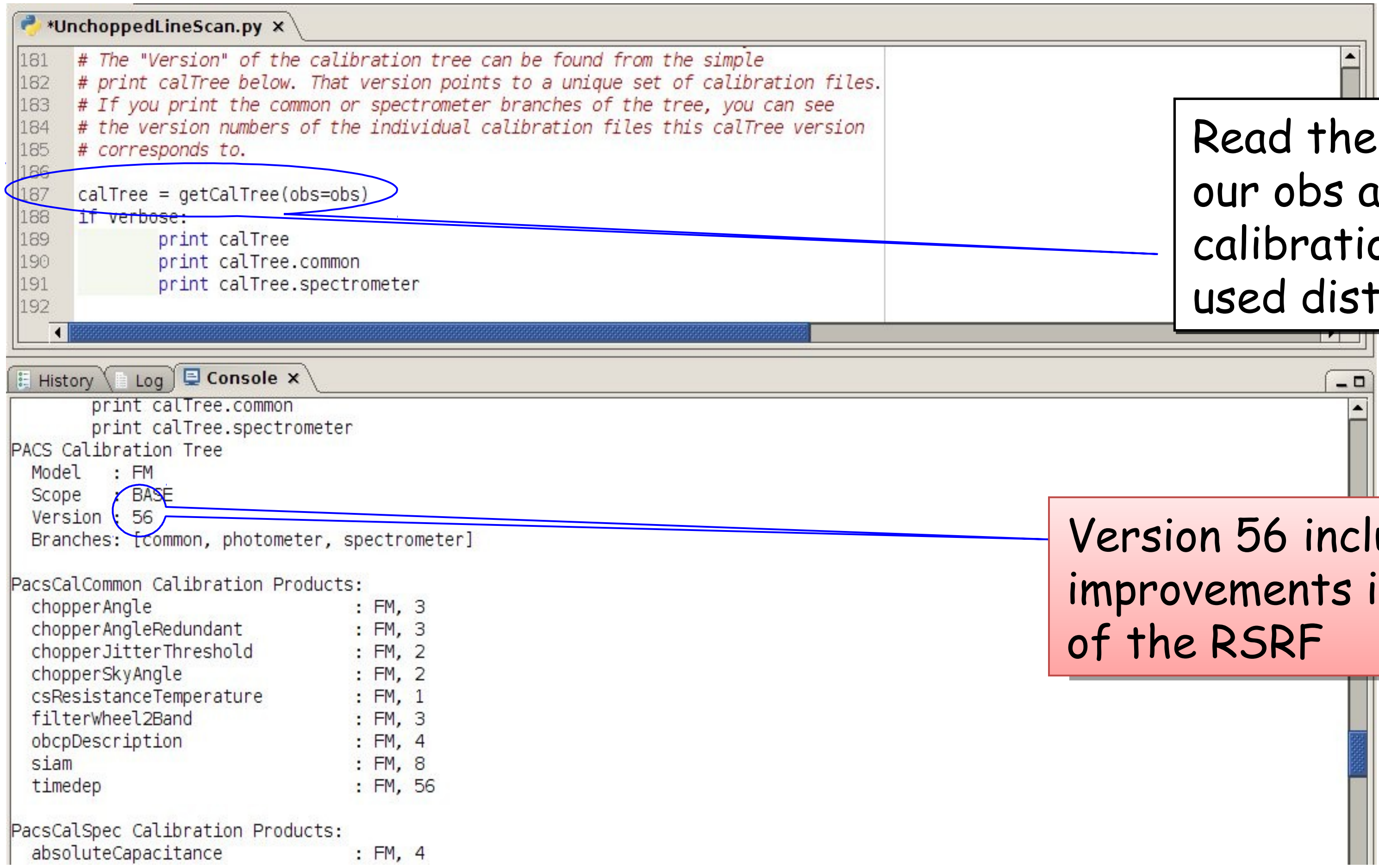
199 of 28106 MB

We select camera = 'red'

Setting the calibration tree

NHSC PACS Web Tutorial

Finally, we set the calibration tree. We can check the calibration used on the archival data with: print obs.meta["calVersion"]



```
# 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.

calTree = getCalTree(obs=obs)
if verbose:
        print calTree
        print calTree.common
        print calTree.spectrometer
```

```
        print calTree.common
        print calTree.spectrometer
PACS Calibration Tree
  Model    : FM
  Scope    : BASE
  Version  : 56
  Branches: [common, photometer, spectrometer]

PacsCalCommon Calibration Products:
  chopperAngle                        : FM, 3
  chopperAngleRedundant               : FM, 3
  chopperJitterThreshold              : FM, 2
  chopperSkyAngle                     : FM, 2
  csResistanceTemperature             : FM, 1
  filterWheel2Band                    : FM, 3
  obcpDescription                     : FM, 4
  siam                                : FM, 8
  timedep                             : FM, 56

PacsCalSpec Calibration Products:
  absoluteCapacitance                 : FM, 4
```

Read the time stamp of our obs and apply the calibration from the used distribution.

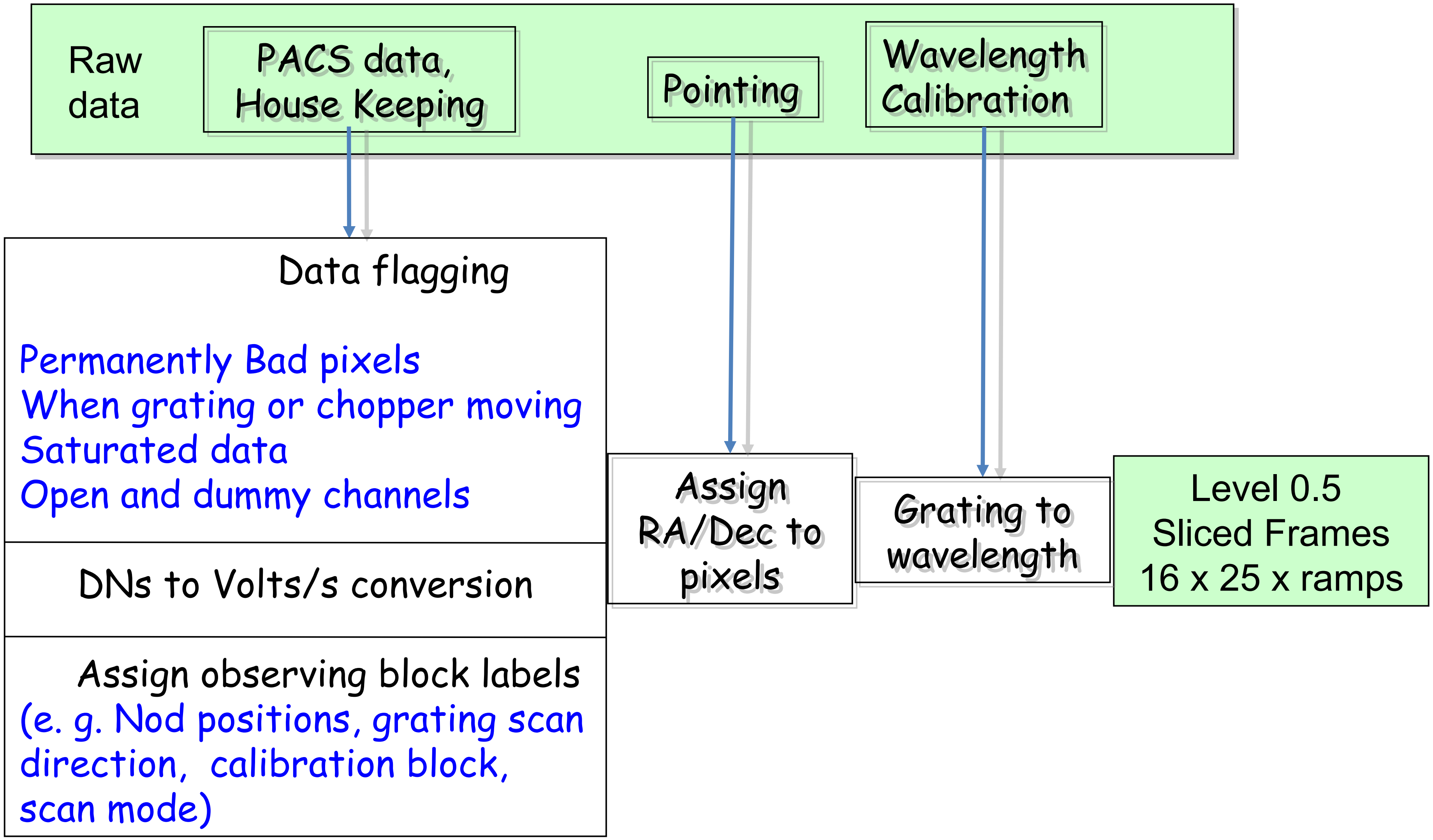
Version 56 includes some improvements in the definition of the RSRF

Step 3

Run the 0 → 0.5 pipeline

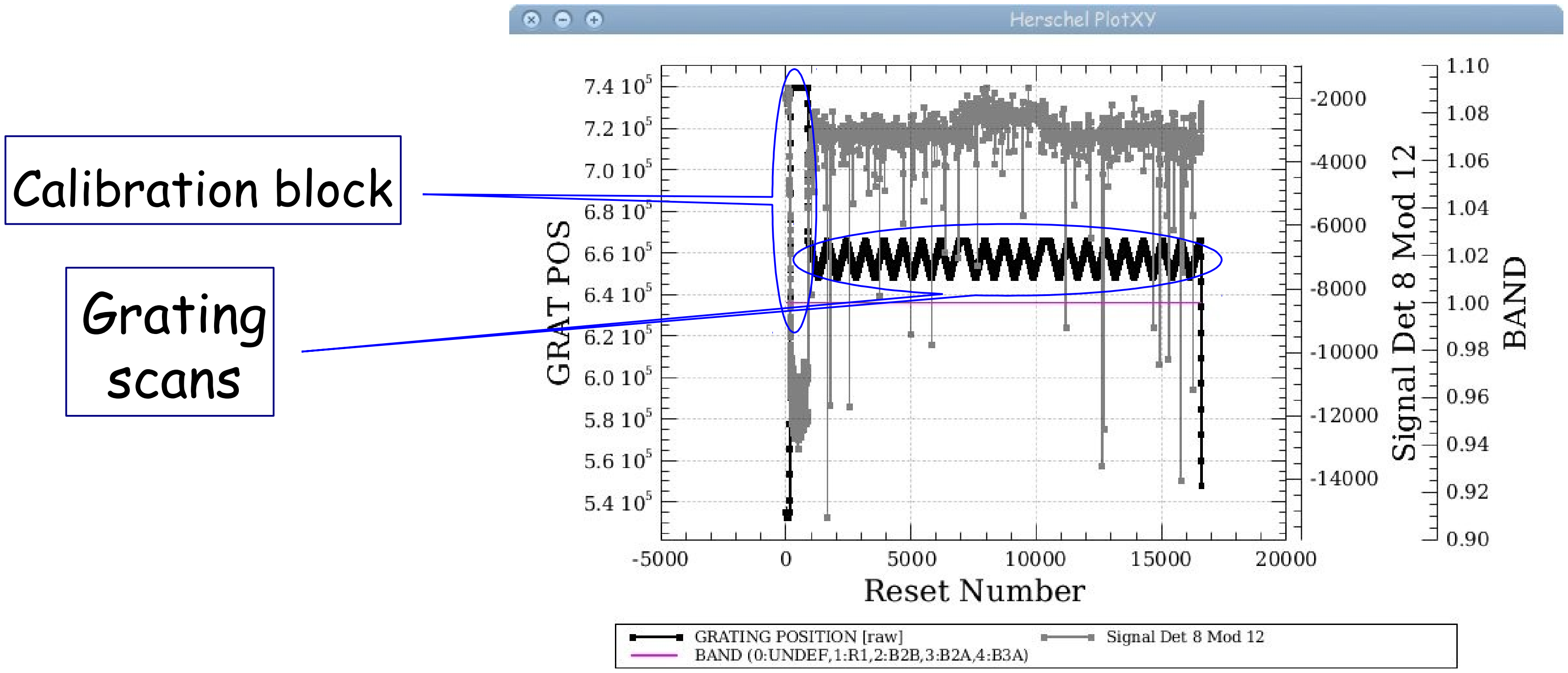
Basic calibration (pointing, wavelength calibration, slicing)

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

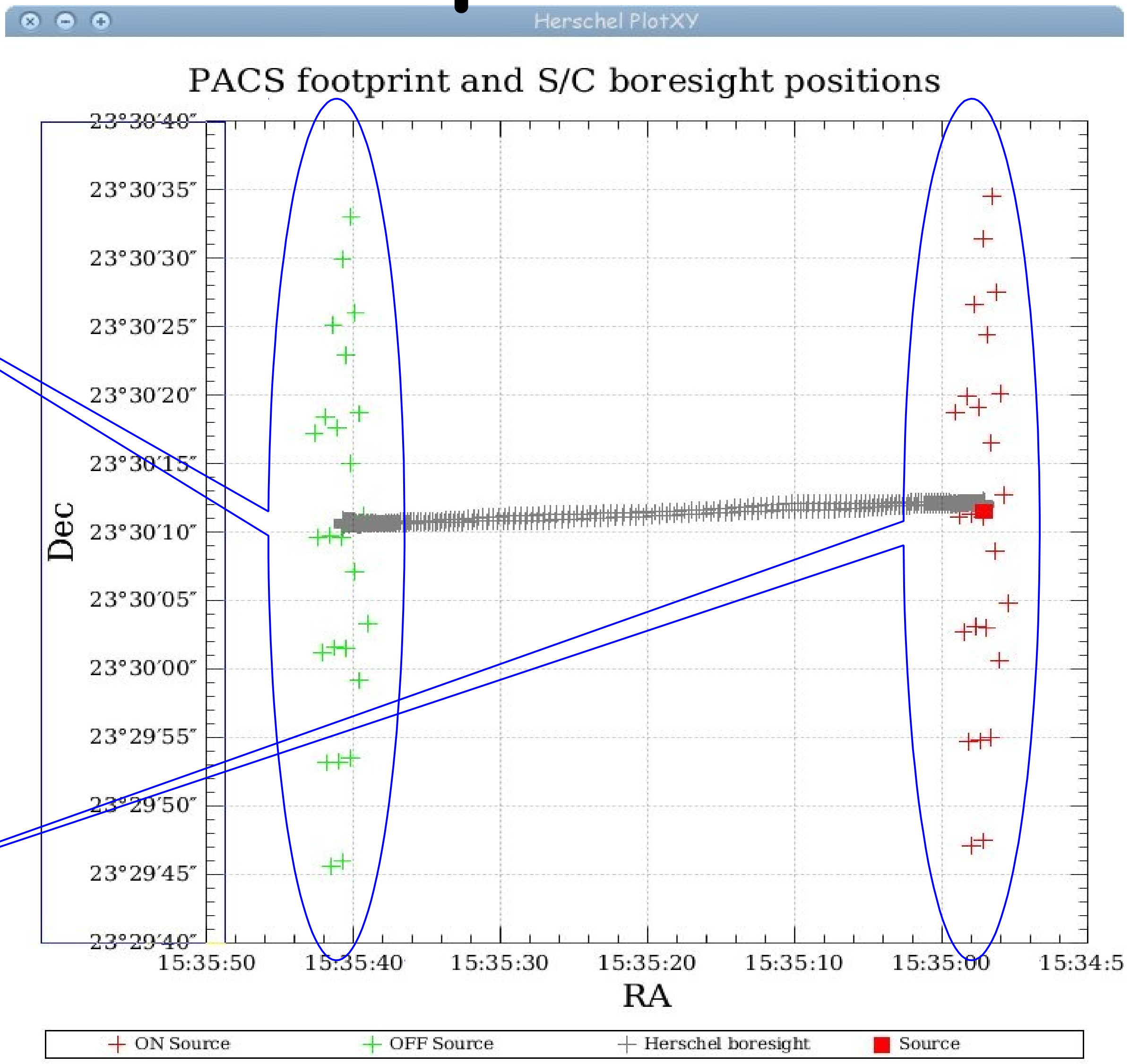
Check: level 0

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

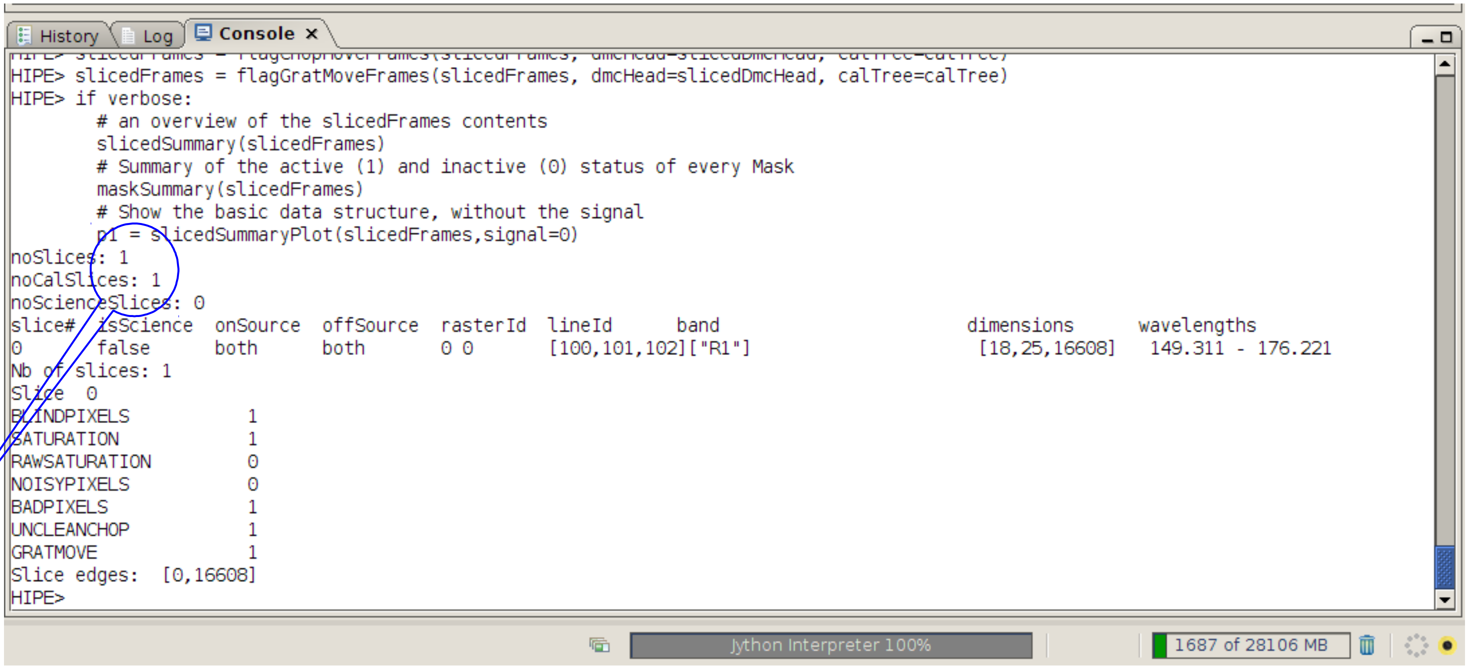


OFF

ON

nhsc.ipac.caltech.edu/helpdesk

Check: before slicing

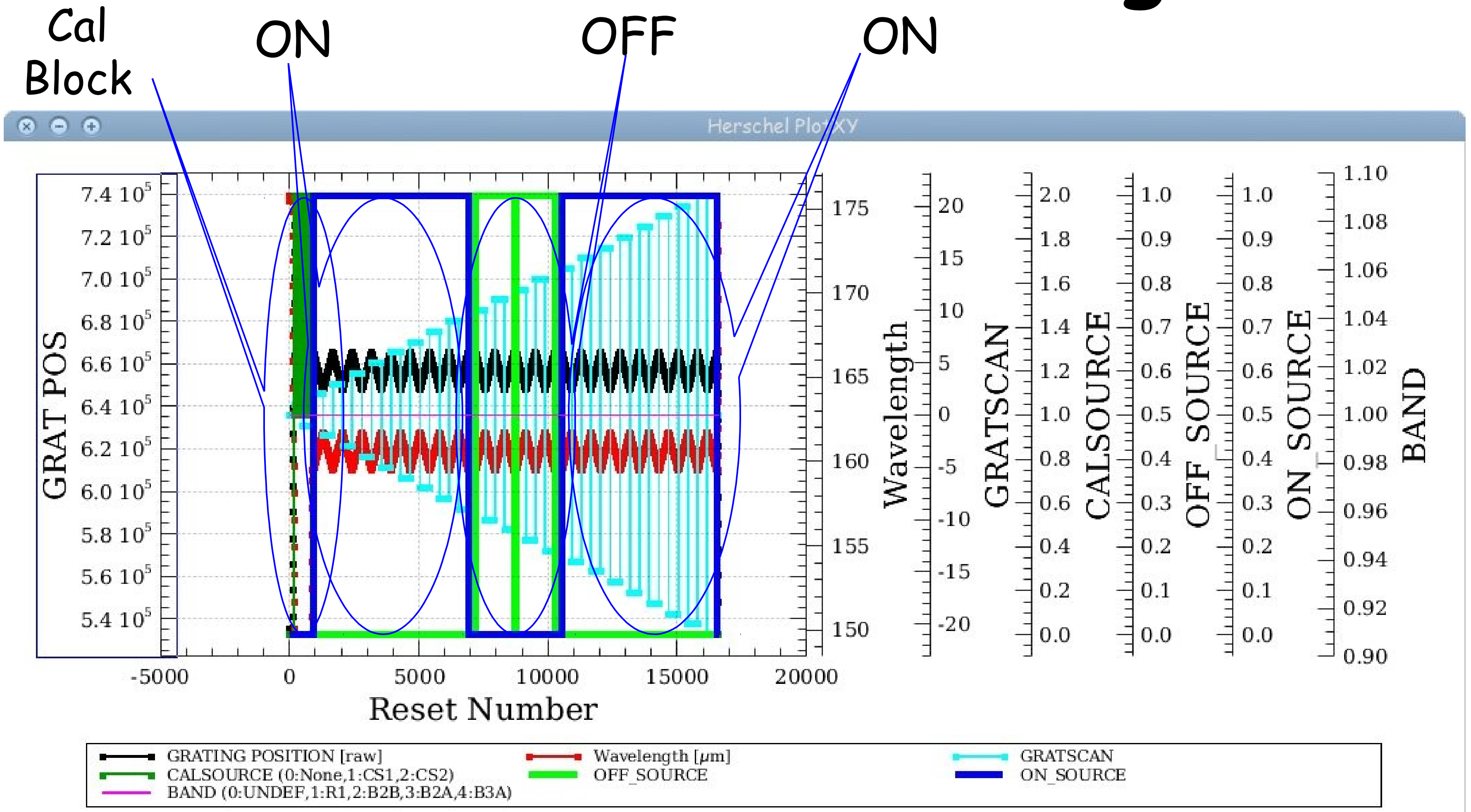


```
HIPE> slicedFrames = flagGratMoveFrames(slicedFrames, dmcHead=slicedDmcHead, calTree=calTree)
HIPE> if verbose:
        # an overview of the slicedFrames contents
        slicedSummary(slicedFrames)
        # Summary of the active (1) and inactive (0) status of every Mask
        maskSummary(slicedFrames)
        # 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

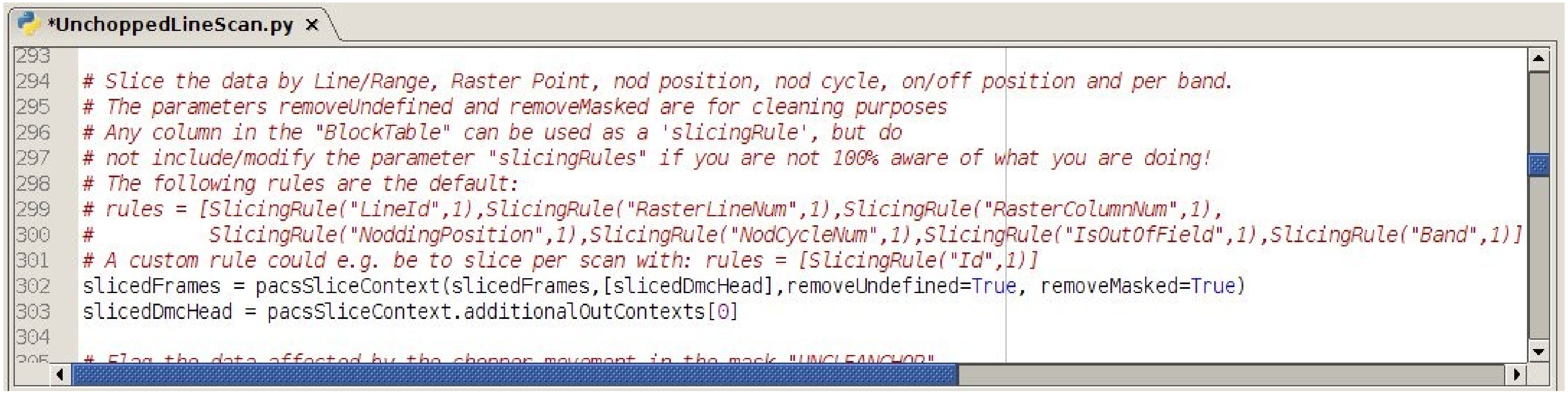
NHSC PACS Web Tutorial



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

nhsc.ipac.caltech.edu/helpdesk

Slicing



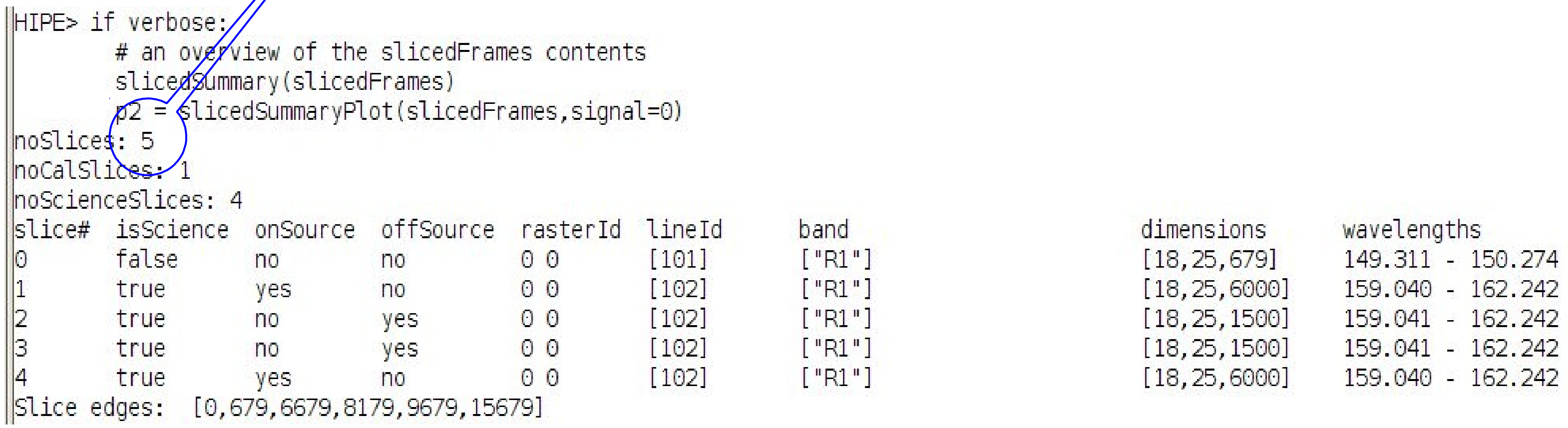
*UnchoppedLineScan.py ×

```
# Slice the data by Line/Range, Raster Point, nod position, nod cycle, on/off position and per band.
# The parameters removeUndefined and removeMasked are for cleaning purposes
# Any column in the "BlockTable" can be used as a 'slicingRule', but do
# not include/modify the parameter "slicingRules" if you are not 100% aware of what you are doing!
# The following rules are the default:
# rules = [SlicingRule("LineId",1),SlicingRule("RasterLineNum",1),SlicingRule("RasterColumnNum",1),
#          SlicingRule("NoddingPosition",1),SlicingRule("NodCycleNum",1),SlicingRule("IsOutOfField",1),SlicingRule("Band",1)]
# A custom rule could e.g. be to slice per scan with: rules = [SlicingRule("Id",1)]
slicedFrames = pacsSliceContext(slicedFrames,[slicedDmcHead],removeUndefined=True, removeMasked=True)
slicedDmcHead = pacsSliceContext.additionalOutContexts[0]

# Flag the data affected by the chopper movement in the mask "UNCLEANCHOP"
```

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 !



```
HIPE> if verbose:
        # an overview of the slicedFrames contents
        slicedSummary(slicedFrames)
        p2 = slicedSummaryPlot(slicedFrames,signal=0)
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]
```

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.

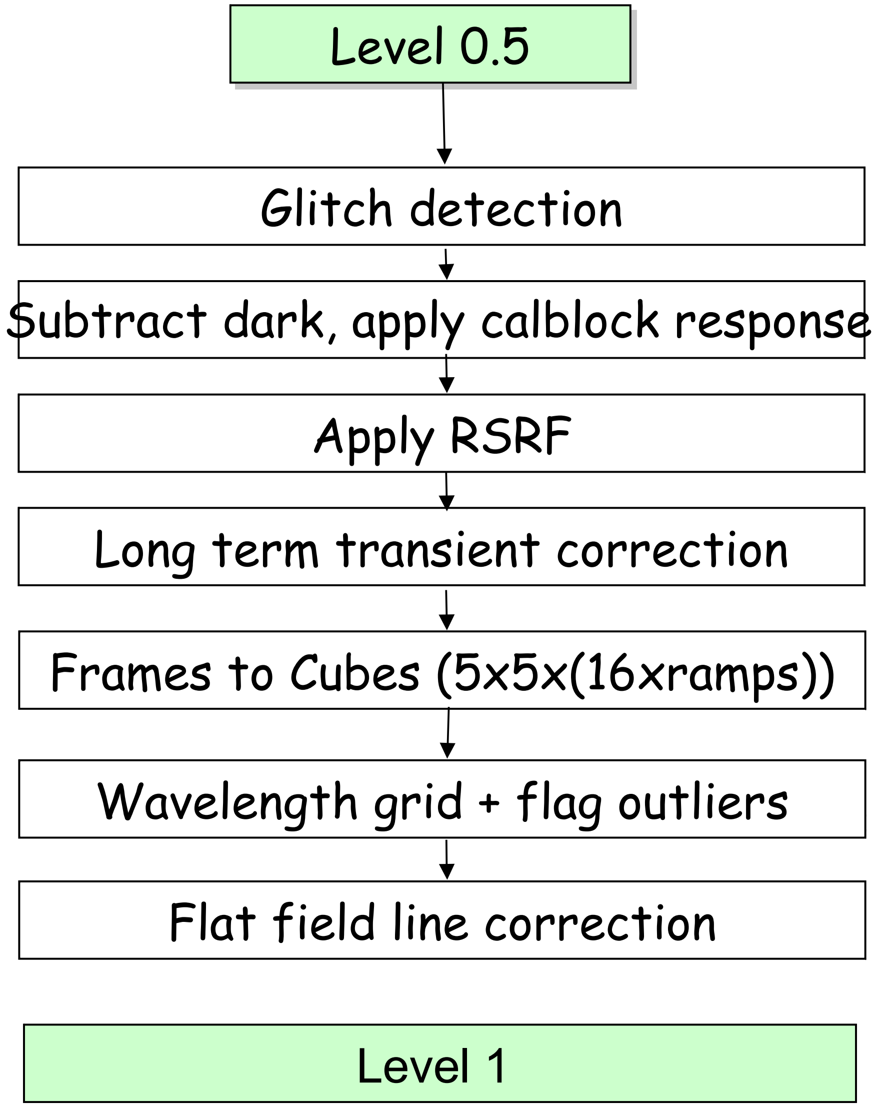
Step 4

Run the 0.5 → 1 pipeline

Glitch detection, chop differentiation, RSRF, flat

NHSC PACS Web Tutorial

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

NASA Herschel Science Center

Glitch detection

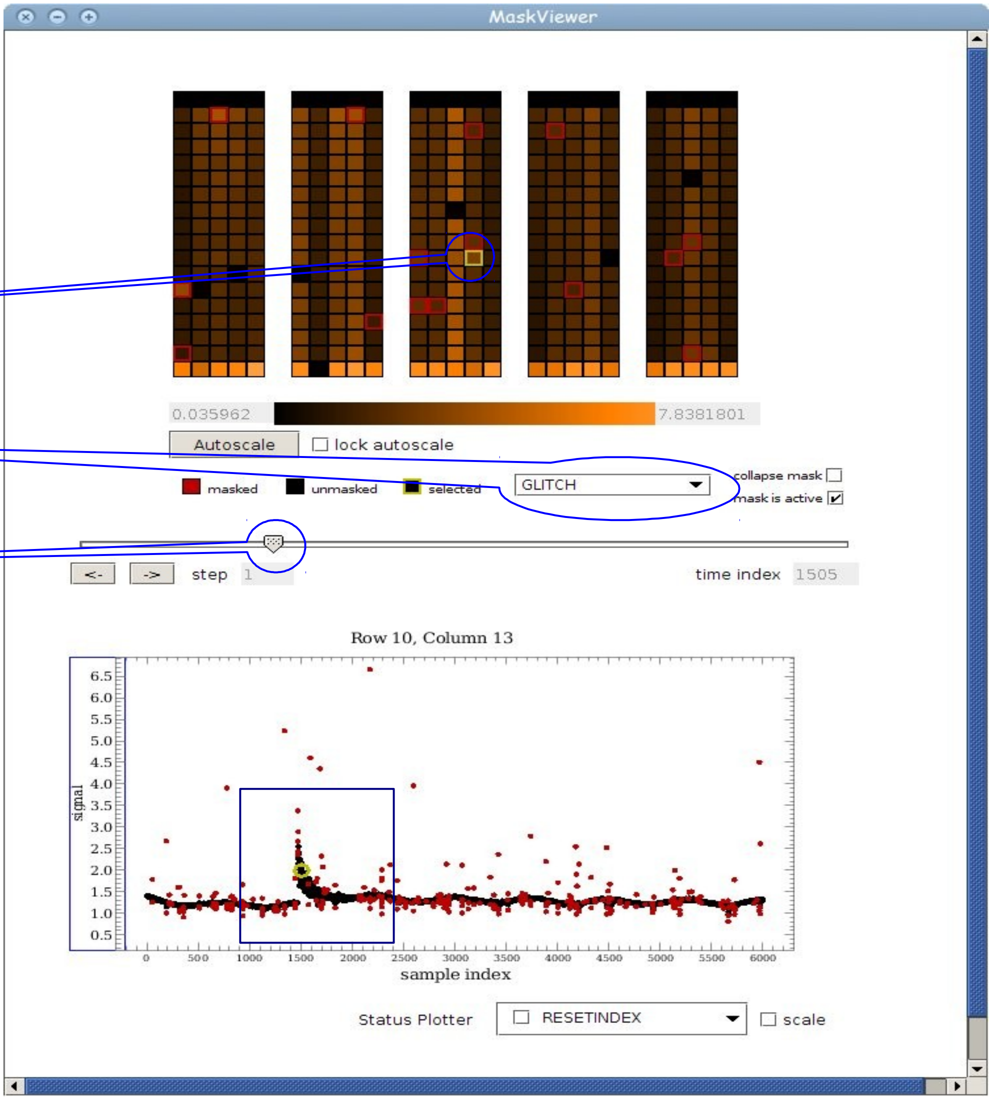
NHSC PACS Web Tutorial

You can view the signal and the masked readouts with the maskviewer. In this case, we find a transient unmasked after a strong cosmic.

Select a pixel by clicking on it

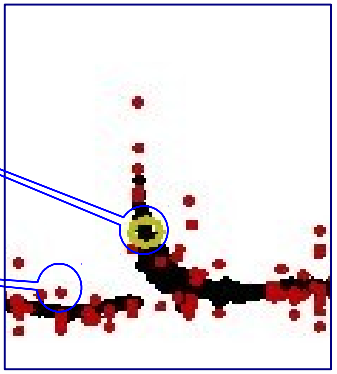
Select a mask

Select a frame



Current frame

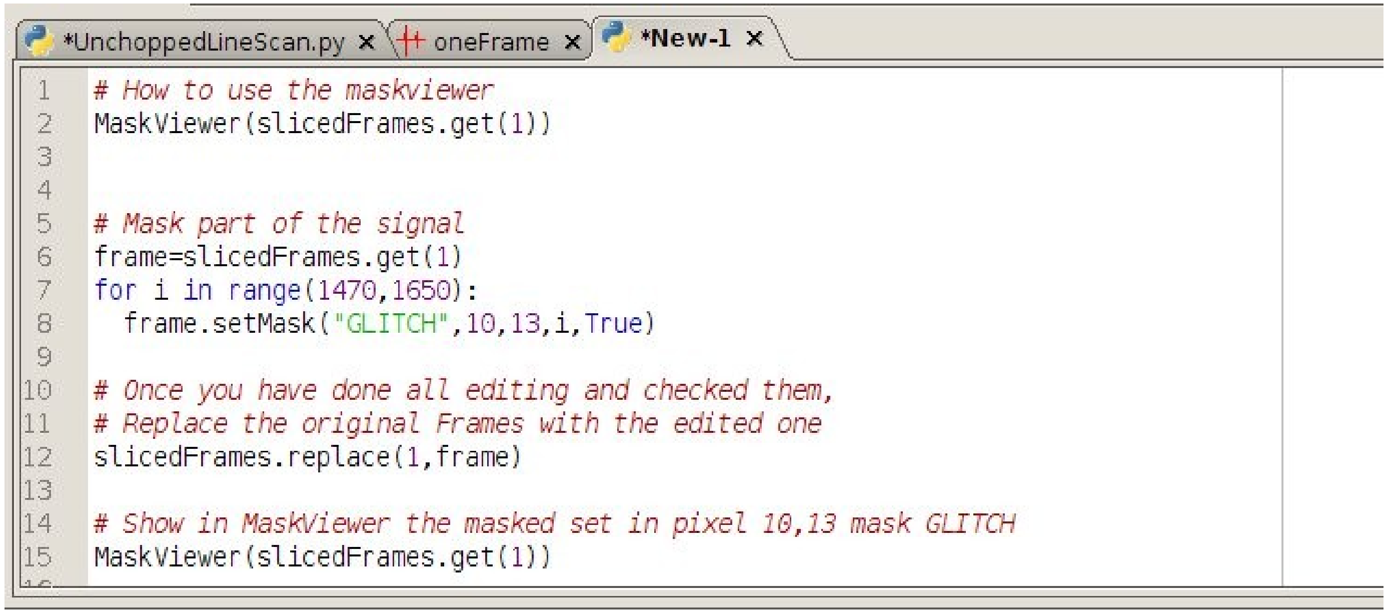
Masked glitch



nhsc.ipac.caltech.edu/helpdesk

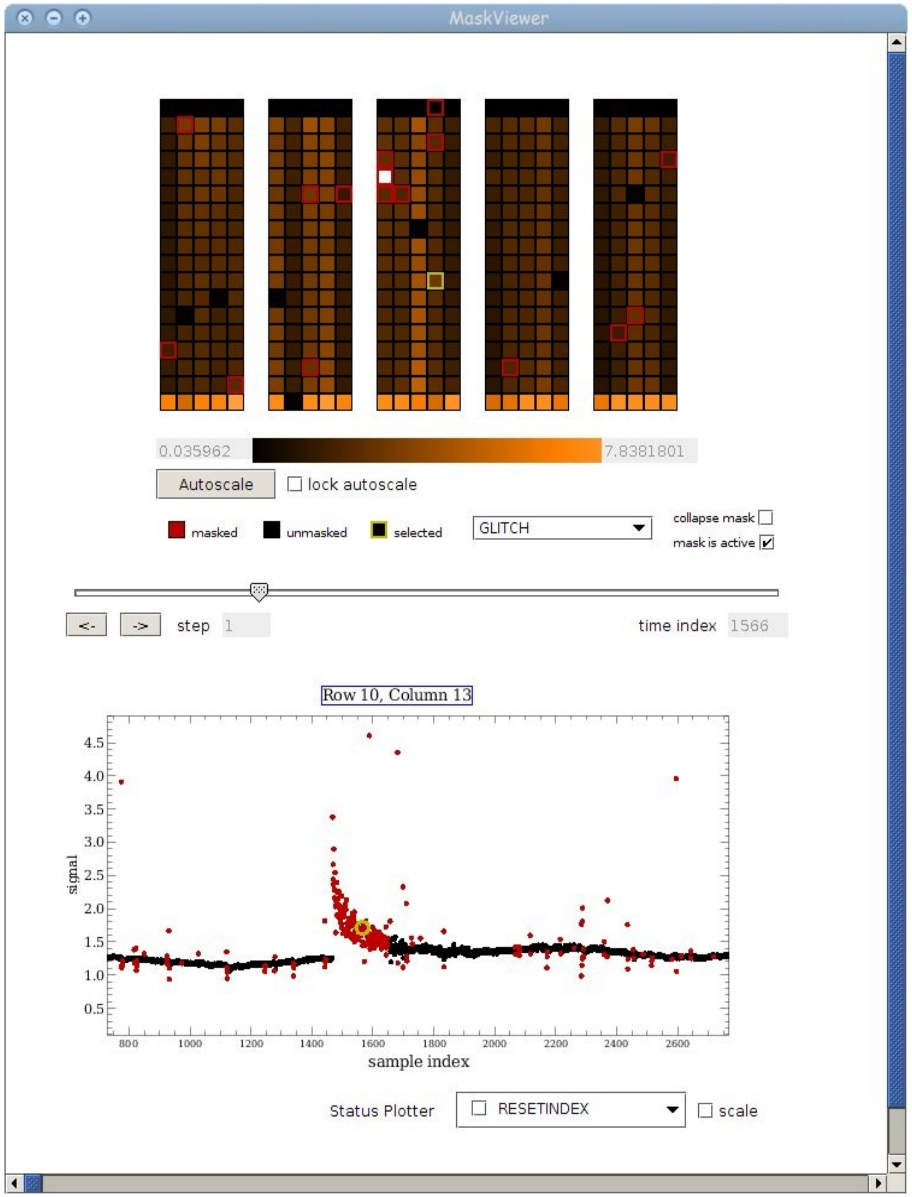
NHSC PACS Web Tutorial

Manual masking

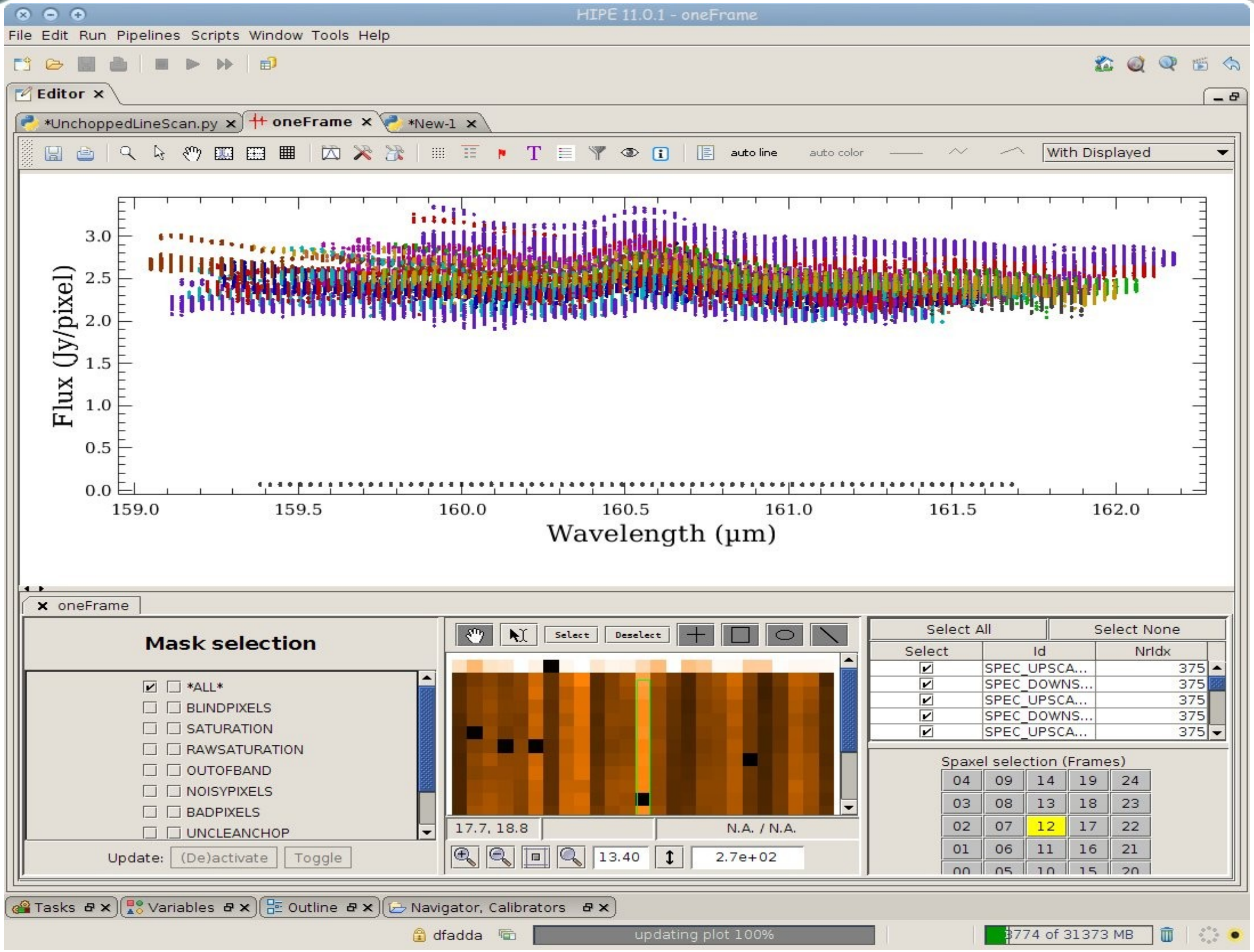


```
# How to use the maskviewer
MaskViewer(slicedFrames.get(1))


# Mask part of the signal
frame=slicedFrames.get(1)
for i in range(1470,1650):
  frame.setMask("GLITCH",10,13,i,True)

# Once you have done all editing and checked them,
# Replace the original Frames with the edited one
slicedFrames.replace(1,frame)

# Show in MaskViewer the masked set in pixel 10,13 mask GLITCH
MaskViewer(slicedFrames.get(1))
```

At this point, we can manually mask part of the signal that are compromised, like the one seen with the maskViewer.
We can open a new tab to write these instructions. Part of the first frame is masked and then checked again with the maskViewer.



The affected region has been masked and it will not be considered in the following.



Select	Id	NrIdx
✓	SPEC_UPSCA...	375
✓	SPEC_DOWNS...	375
✓	SPEC_UPSCA...	375
✓	SPEC_DOWNS...	375
✓	SPEC_UPSCA...	375

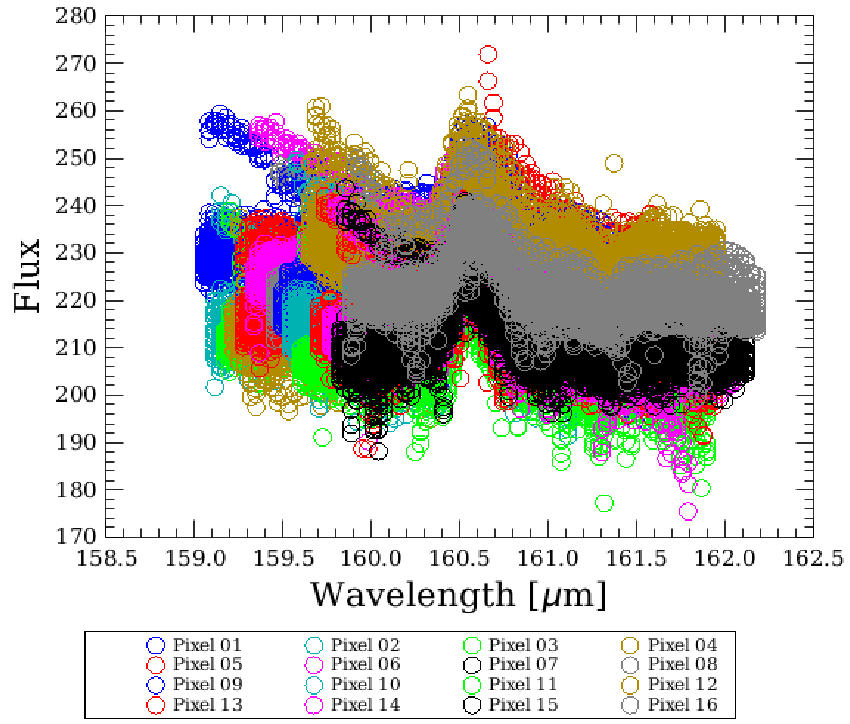
Pixels can be now examined with the Spectrum Explorer

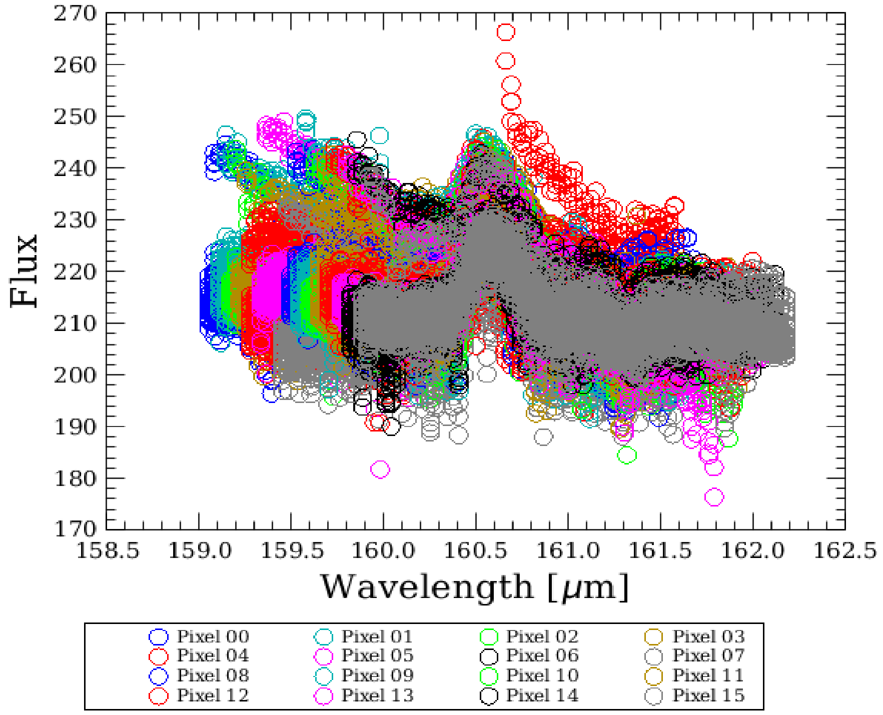
nhsc.ipac.caltech.edu/helpdesk

NASA Herschel Science Center

RSRF, Dark, Response

NHSC PACS Web Tutorial

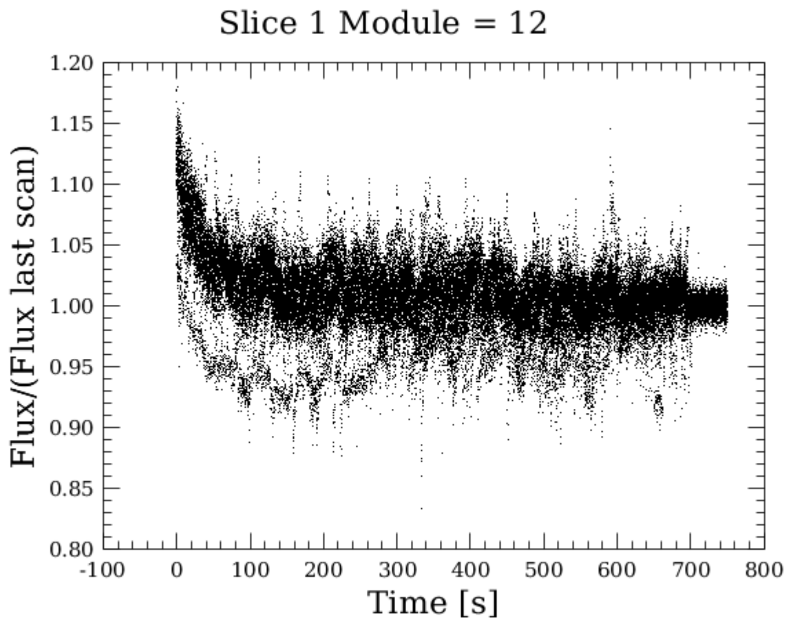


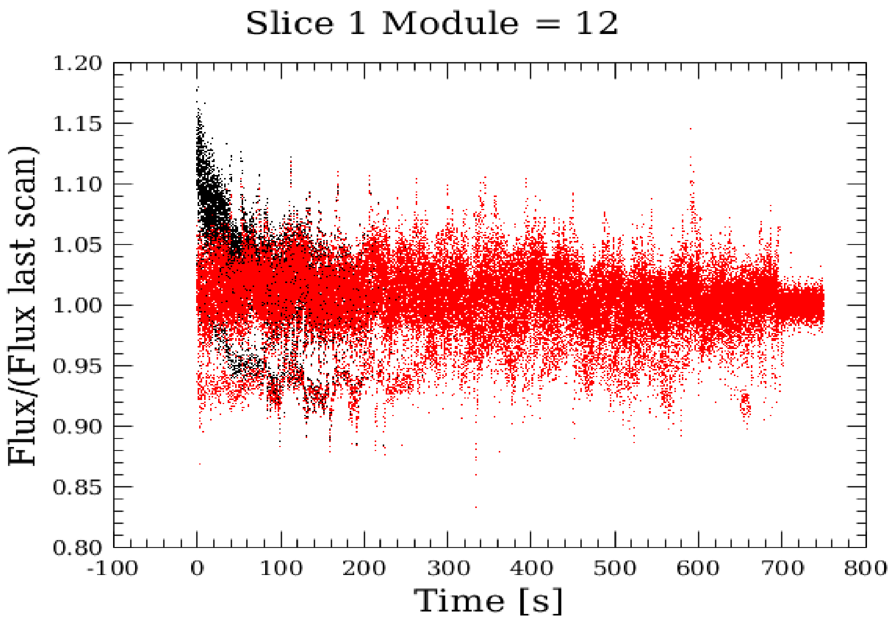


Central module before and after applying several corrections (dark, response, and RSRF)

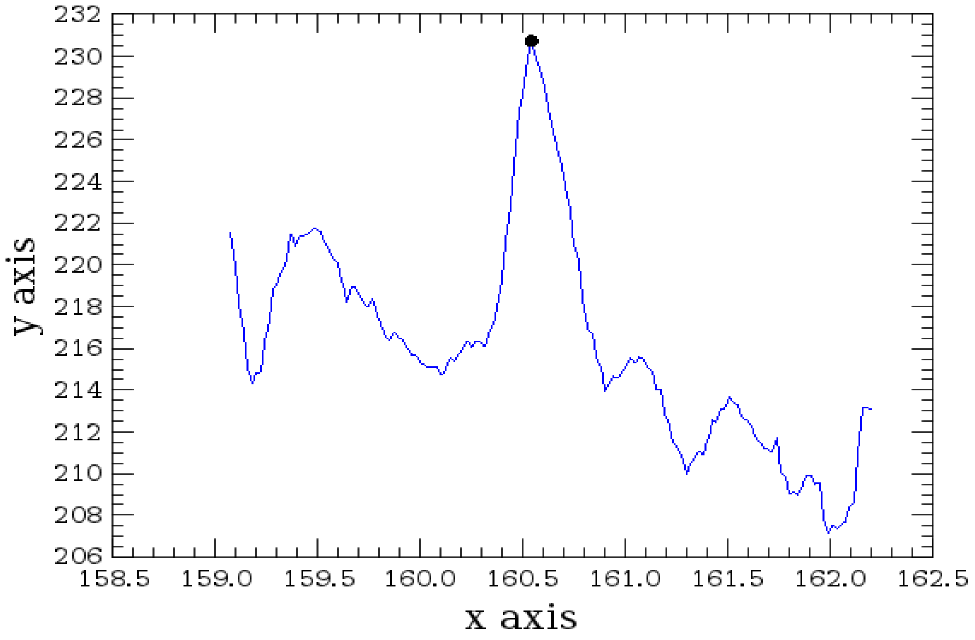
Long term transients

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.





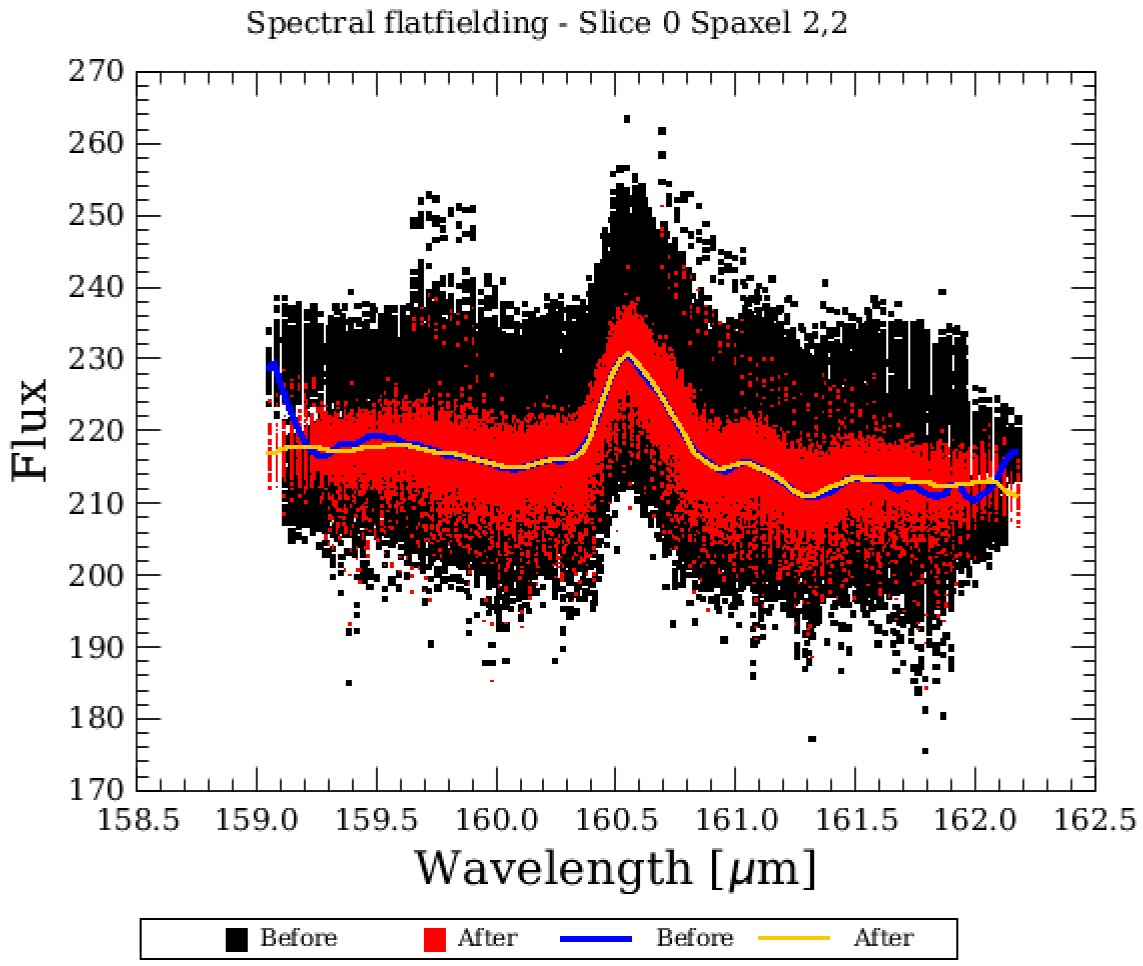
Spectral flat field



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.

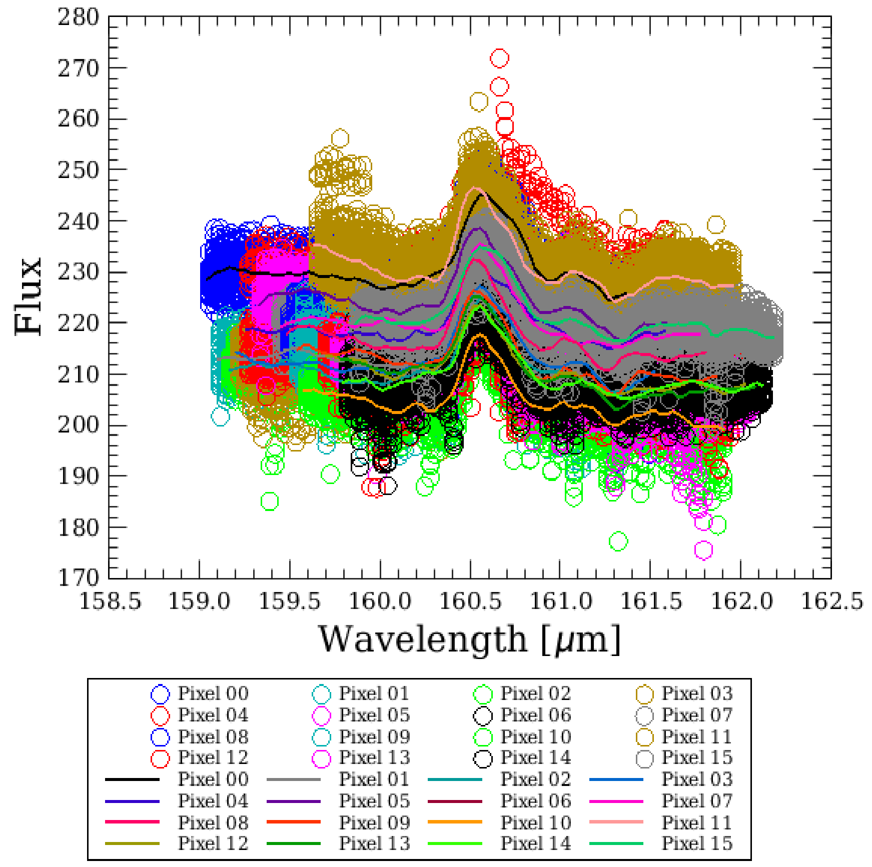
Spectral flat field

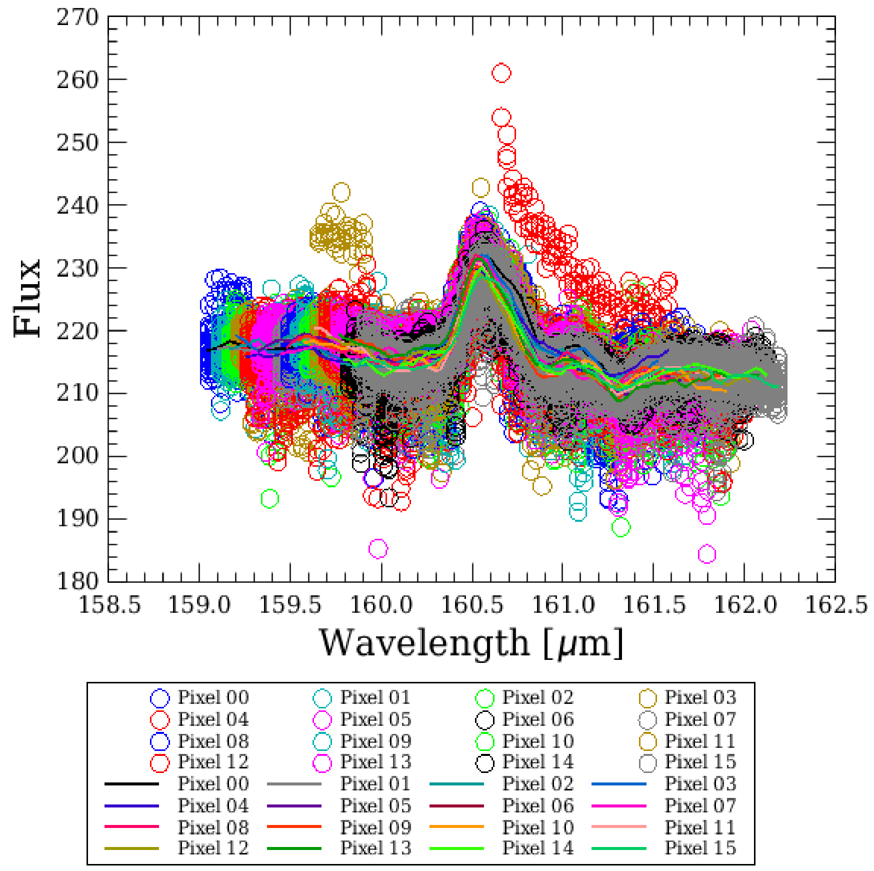
NHSC PACS
Web Tutorial



nhsc.ipac.caltech.edu/helpdesk

Spectral flat field

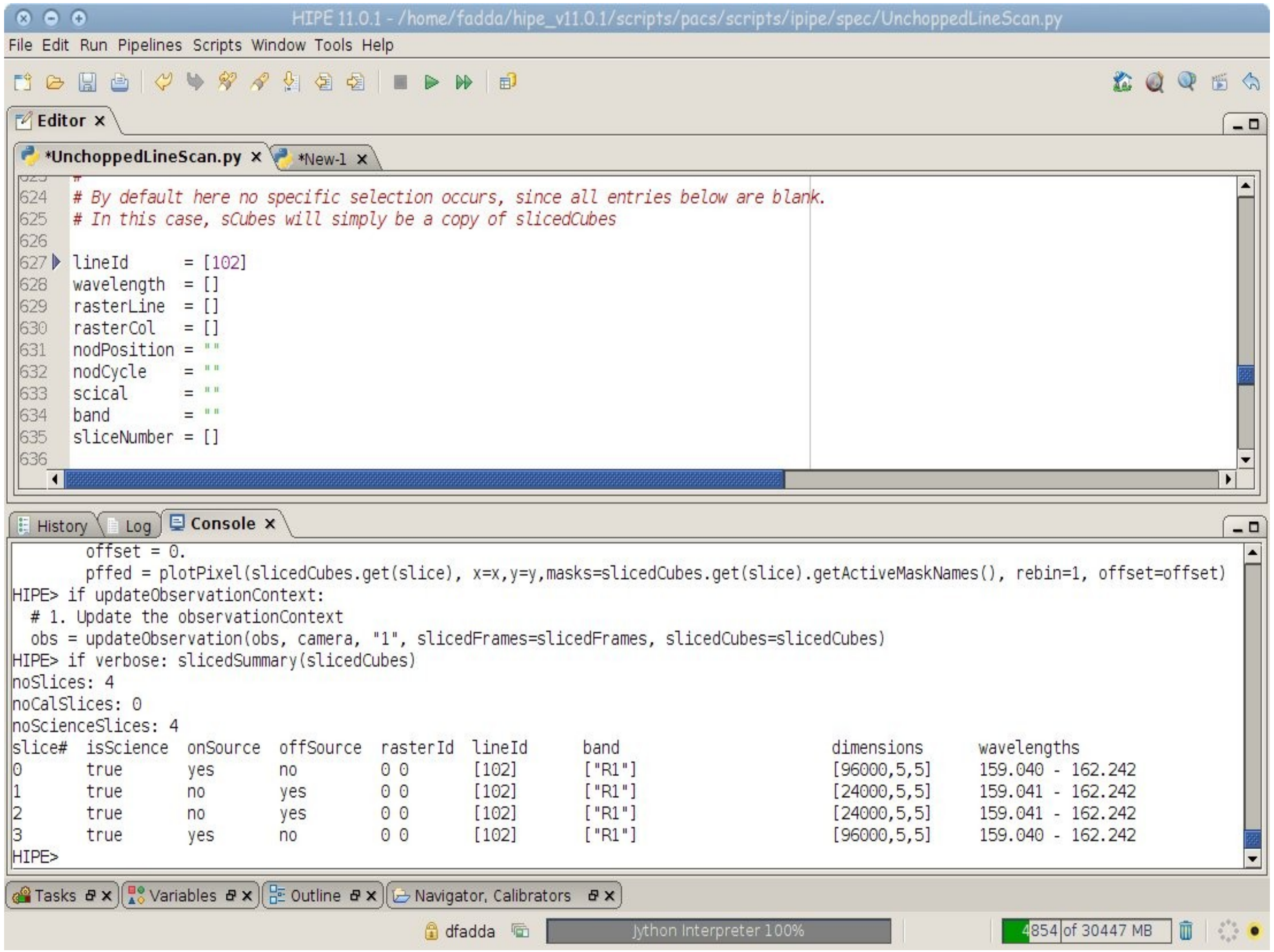




Central module before and after the first spectral flat-fielding
The transient correction has greatly improved the final result.

Level 1 → 2

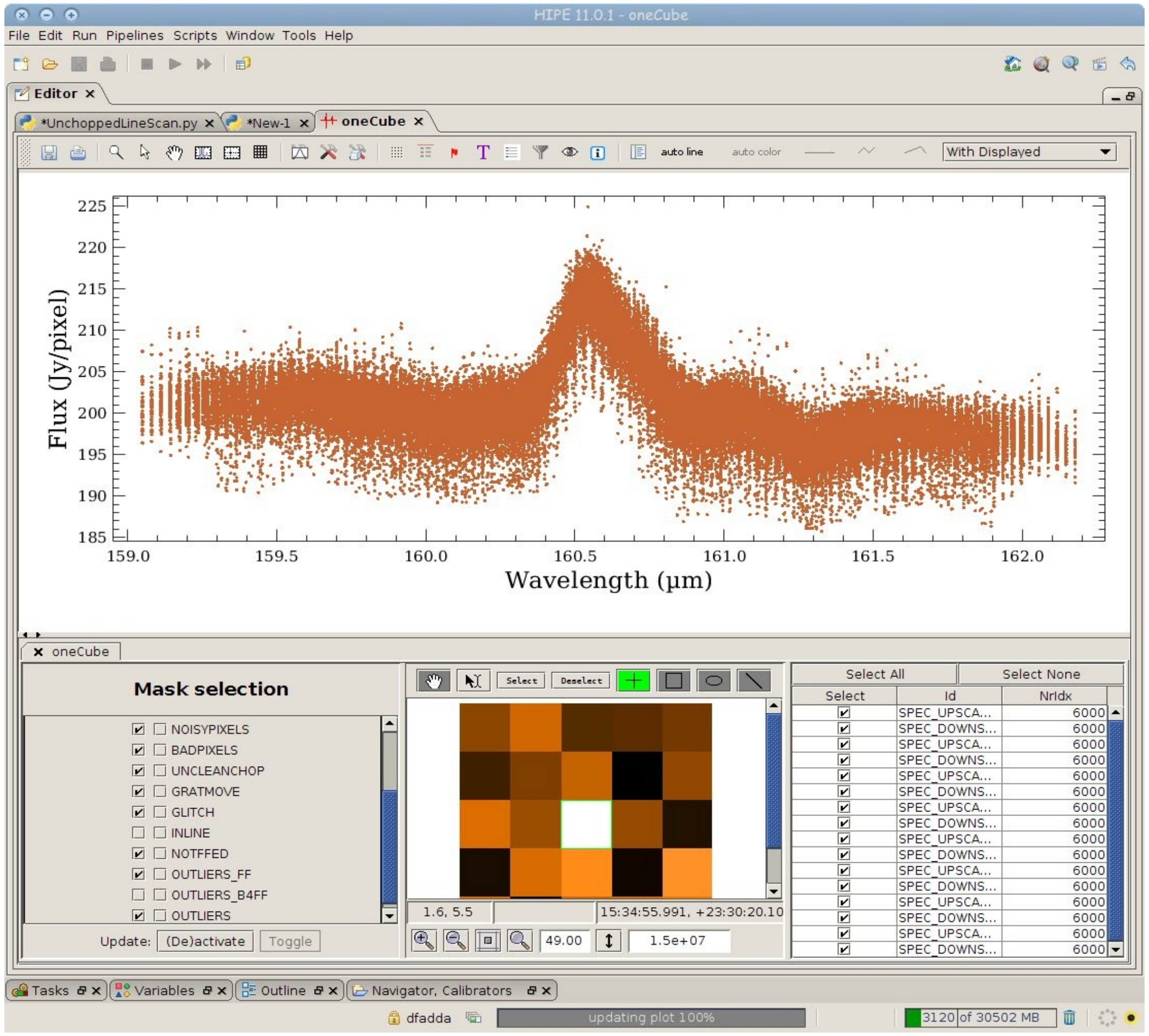
NHSC PACS Web Tutorial


HIPE 11.0.1 - /home/fadda/hipe_v11.0.1/scripts/pacs/scripts/ipipe/spec/UnchoppedLineScan.py
File Edit Run Pipelines Scripts Window Tools Help
Editor
*UnchoppedLineScan.py
*New-1
History
Log
Console
Tasks
Variables
Outline
Navigator, Calibrators
dfadda
Jython Interpreter 100%
4854 of 30447 MB

```
# By default here no specific selection occurs, since all entries below are blank.
# In this case, sCubes will simply be a copy of slicedCubes

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

```
        offset = 0.
        pffed = plotPixel(slicedCubes.get(slice), x=x,y=y,masks=slicedCubes.get(slice).getActiveMaskNames(), rebin=1, offset=offset)
HIPE> if updateObservationContext:
  # 1. Update the observationContext
  obs = updateObservation(obs, camera, "1", slicedFrames=slicedFrames, slicedCubes=slicedCubes)
HIPE> if verbose: slicedSummary(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. To proceed, we choose a line (102) by reading the output of slicedSummary. The level two routines are explained in the PACS 302 tutorial.



Mask selection

- [✓] NOISYPIXELS
- [✓] BADPIXELS
- [✓] UNCLEANCHOP
- [✓] GRATMOVE
- [✓] GLITCH
- [ ] INLINE
- [✓] NOTFFED
- [✓] OUTLIERS_FF
- [ ] OUTLIERS_B4FF
- [✓] OUTLIERS

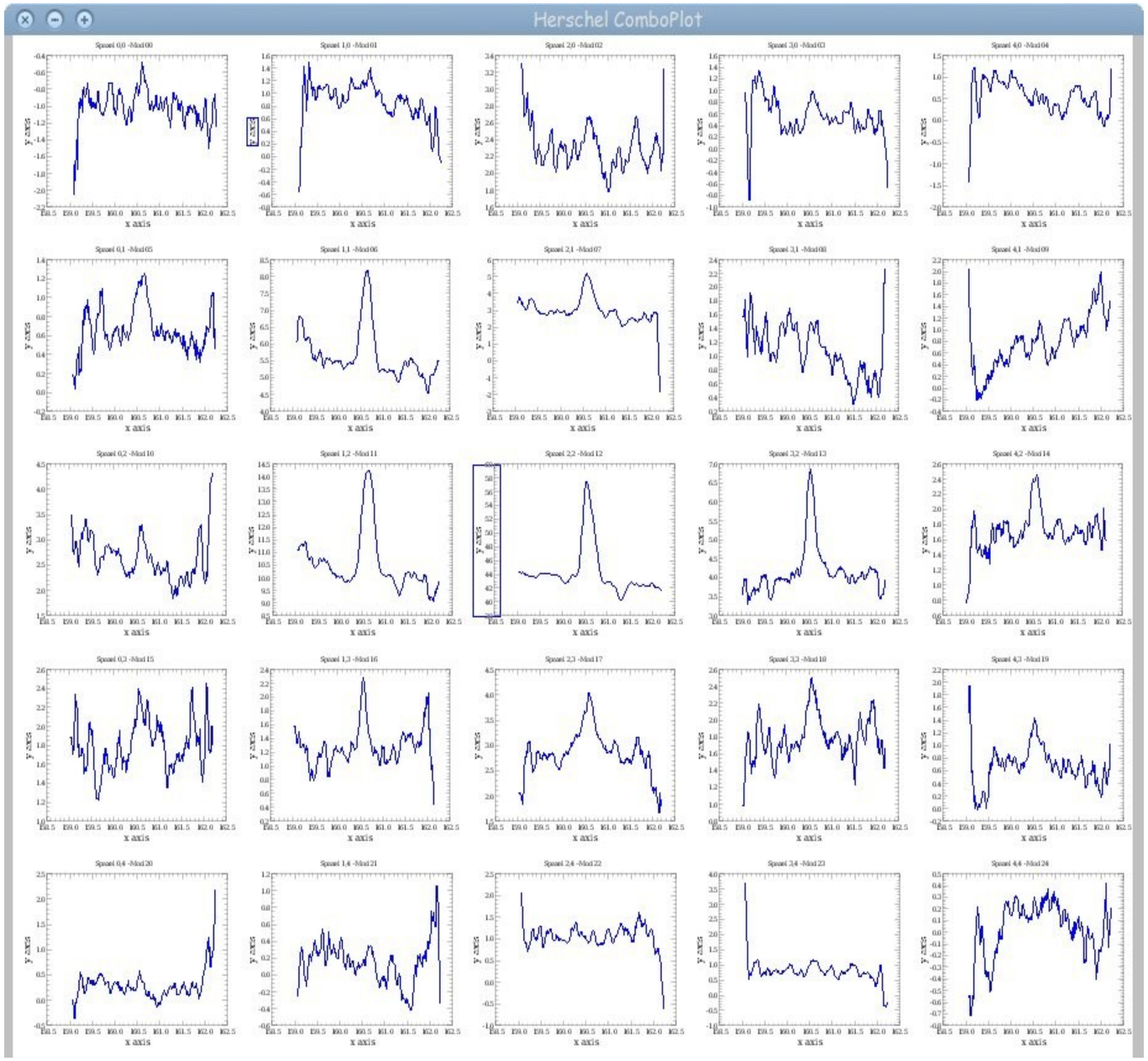
1.6, 5.5 | 15:34:55.991, +23:30:20.10

49.00 | 1.5e+07

Select	Id	NrIdx
✓	SPEC_UPSCA...	6000
✓	SPEC_DOWNS...	6000
✓	SPEC_UPSCA...	6000
✓	SPEC_DOWNS...	6000
✓	SPEC_UPSCA...	6000
✓	SPEC_DOWNS...	6000
✓	SPEC_UPSCA...	6000
✓	SPEC_DOWNS...	6000
✓	SPEC_UPSCA...	6000
✓	SPEC_DOWNS...	6000
✓	SPEC_UPSCA...	6000
✓	SPEC_DOWNS...	6000
✓	SPEC_UPSCA...	6000
✓	SPEC_DOWNS...	6000
✓	SPEC_UPSCA...	6000
✓	SPEC_DOWNS...	6000

3120 of 30502 MB

We can inspect the line with Spectrum Explorer



After subtracting the OFF cube from the ON cube, a multi-plot feature allows one to see all the spectra in the different spaxels.

Multi-threading

When running interactive pipelines, be sure to use the multi-threading option. **This option is only available for the PACS spectrometer** and it will speed-up your reduction by exploiting all the cores of your machine.

Using this option is extraordinarily simple. Just two lines:

```
Configuration.setProperty("herschel.pacs.spg.common.superThreadCount","4")
Configuration.setProperty("herschel.pacs.spg.spec.threadCount","8")
```

Some tasks are threaded. The other ones are naturally threaded by exploiting the slicing of the data.
The **superThreadCount** is used for the general threading, while the **threadCount** is used for the threaded tasks.

The optimal choice of the threading parameters depends on the number of cores on your machine and the number of slices. Memory is not an issue, because the first part of the pipeline is unthreaded and puts the entire data in memory. When data are sliced, the total memory used is always the same.

An automatic choice is done by putting:

```
Configuration.setProperty("herschel.pacs.spg.common.superThreadCount","0")
Configuration.setProperty("herschel.pacs.spg.spec.threadCount","0")
```

Otherwise, a good choice is to put threadCount equal to the number of cores and superThreadCount equal to the number of slices.

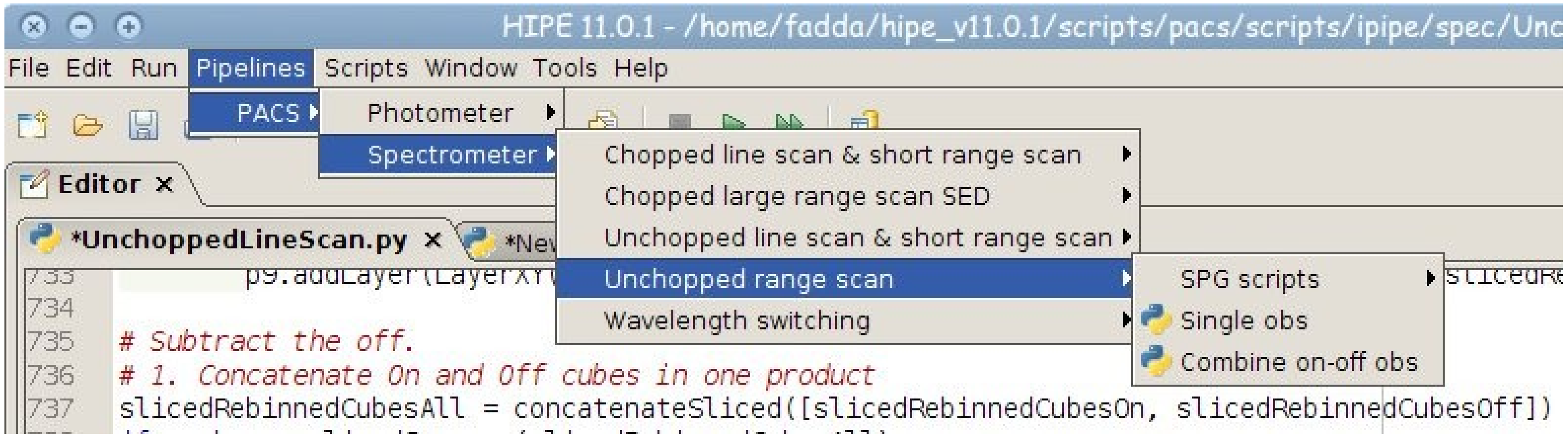
Unchopped range scan

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 11.0.1 - /home/fadda/hipe_v11.0.1/scripts/pacs/scripts/ipipe/spec/Unc
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
SPG scripts
Single obs
Combine on-off obs
Editor
*UnchoppedLineScan.py
*New

```
        p9.addLayer(LayerXY

# Subtract the off.
# 1. Concatenate On and Off cubes in one product
slicedRebinnedCubesAll = concatenateSliced([slicedRebinnedCubesOn, slicedRebinnedCubesOff])
```

The two interactive scripts available to reduce range unchopped observations:
a) Single obs for the ON and OFF observations
b) Combine the two AORs.