NHSC/PACS Web Tutorials

Running the PACS Spectrometer pipeline for CHOP/NOD Mode

PACS-301

Pipeline Level 0 to 1 processing

Prepared by Dario Fadda

Updated by Babar Ali, February 2013

Updated by Steve Lord, Oct 2013

Updated for HIPE 12.0.0 by David Shupe, May 2014

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 chop/nod mode.

Pre-requisites

The following tutorials should be read before this one:

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

Sequel: *PACS-302* – Level 1-2 processing

Overview

Step 1 Check HIPE version and your local memory

Step 2 Set up script for the particular Obs ID

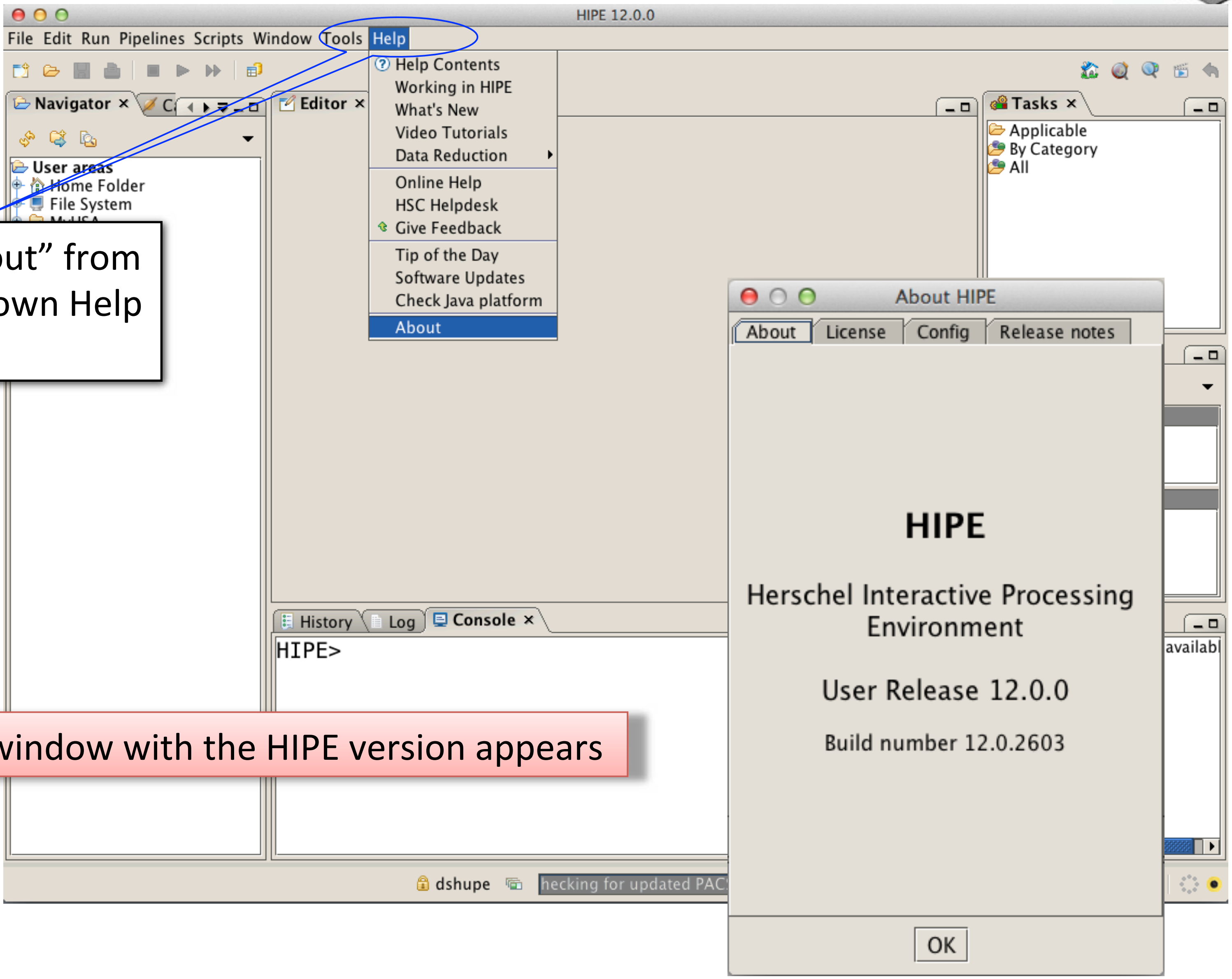
Step 3 Run the 0 → 0.5 pipeline

Step 4 Run the 0.5 → 1 pipeline

Step 1

Check HIPE version and memory allocation

The version used for the tutorial is User Release 12.0.0, also known as Build number 12.0.2603



```
HIPE>
```

HIPE

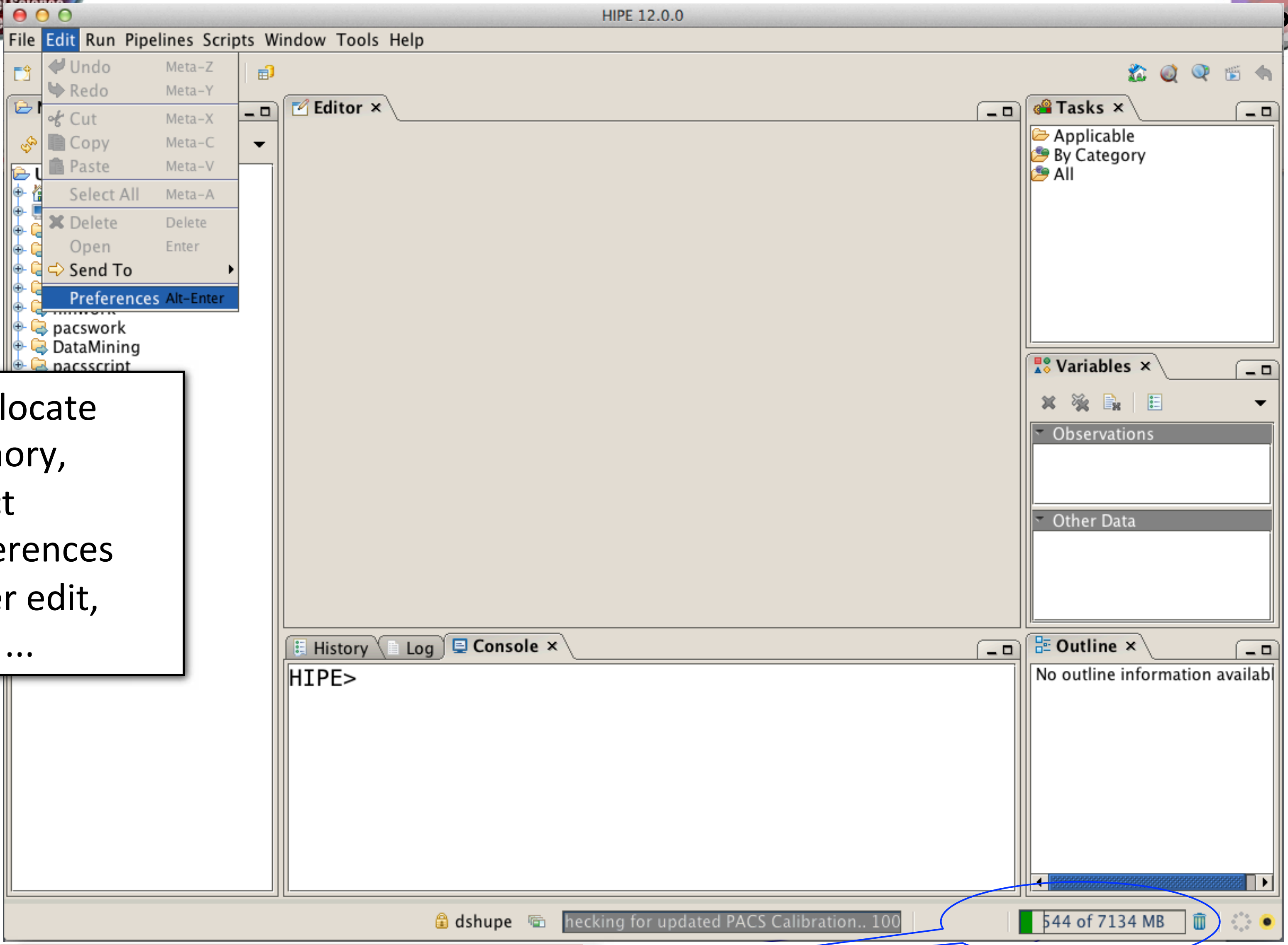
Herschel Interactive Processing Environment

User Release 12.0.0

Build number 12.0.2603

Select “about” from the drop down Help menu

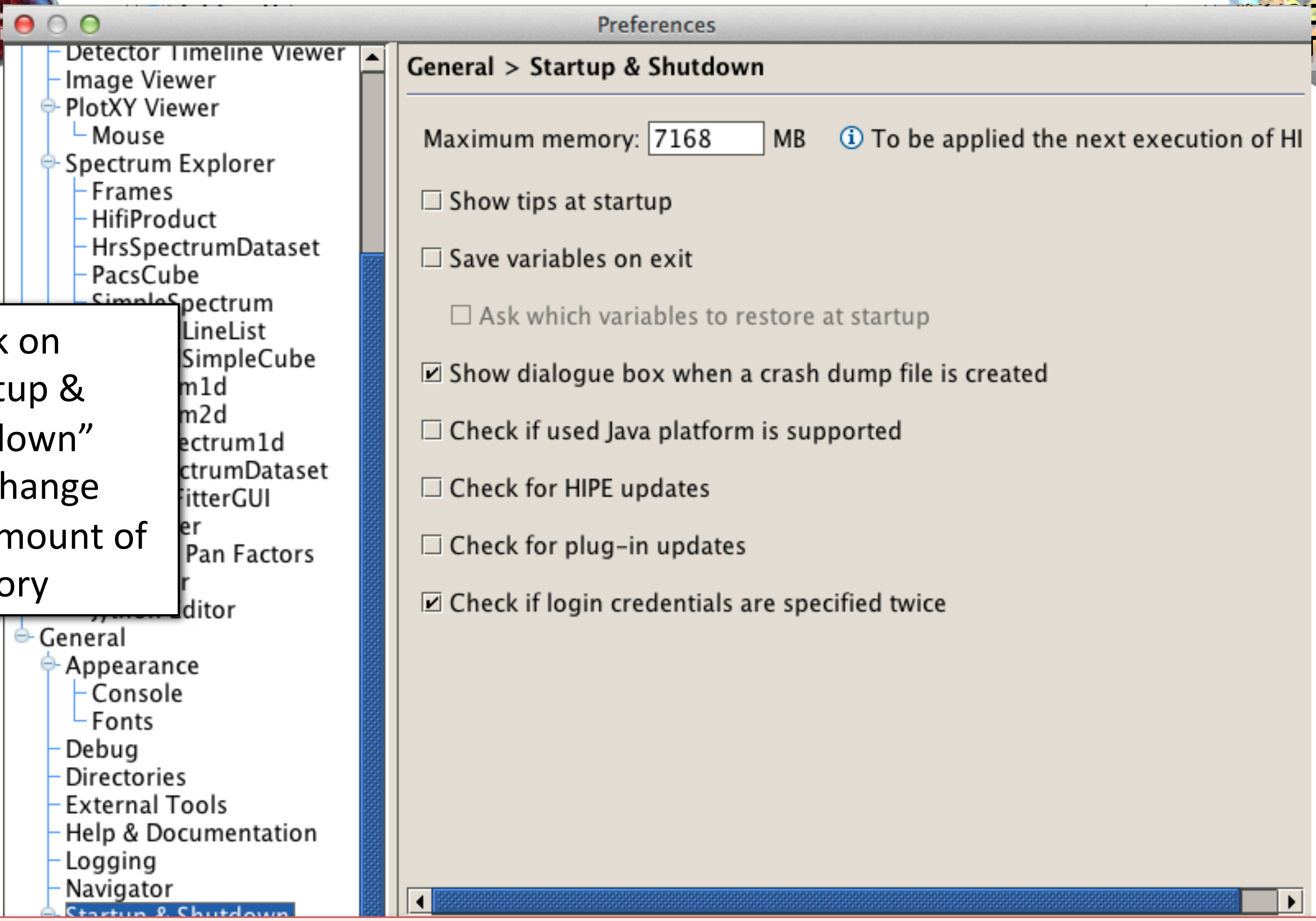
A pop-up window with the HIPE version appears



```
HIPE>
```

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

N.b.:, Memory used and available



General > Startup & Shutdown

Maximum memory: 7168 MB ⓘ To be applied the next execution of HI

☐ 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

…click on "Startup & Shutdown" and change the amount of memory

The allocated memory should be a bit smaller than the total RAM of your computer. (e.g. 7.5 out of 8.0 Gbytes)
You must exit and restart HIPE to obtain the new amount of memory.

nhsc.ipac.caltech.edu/helpdesk

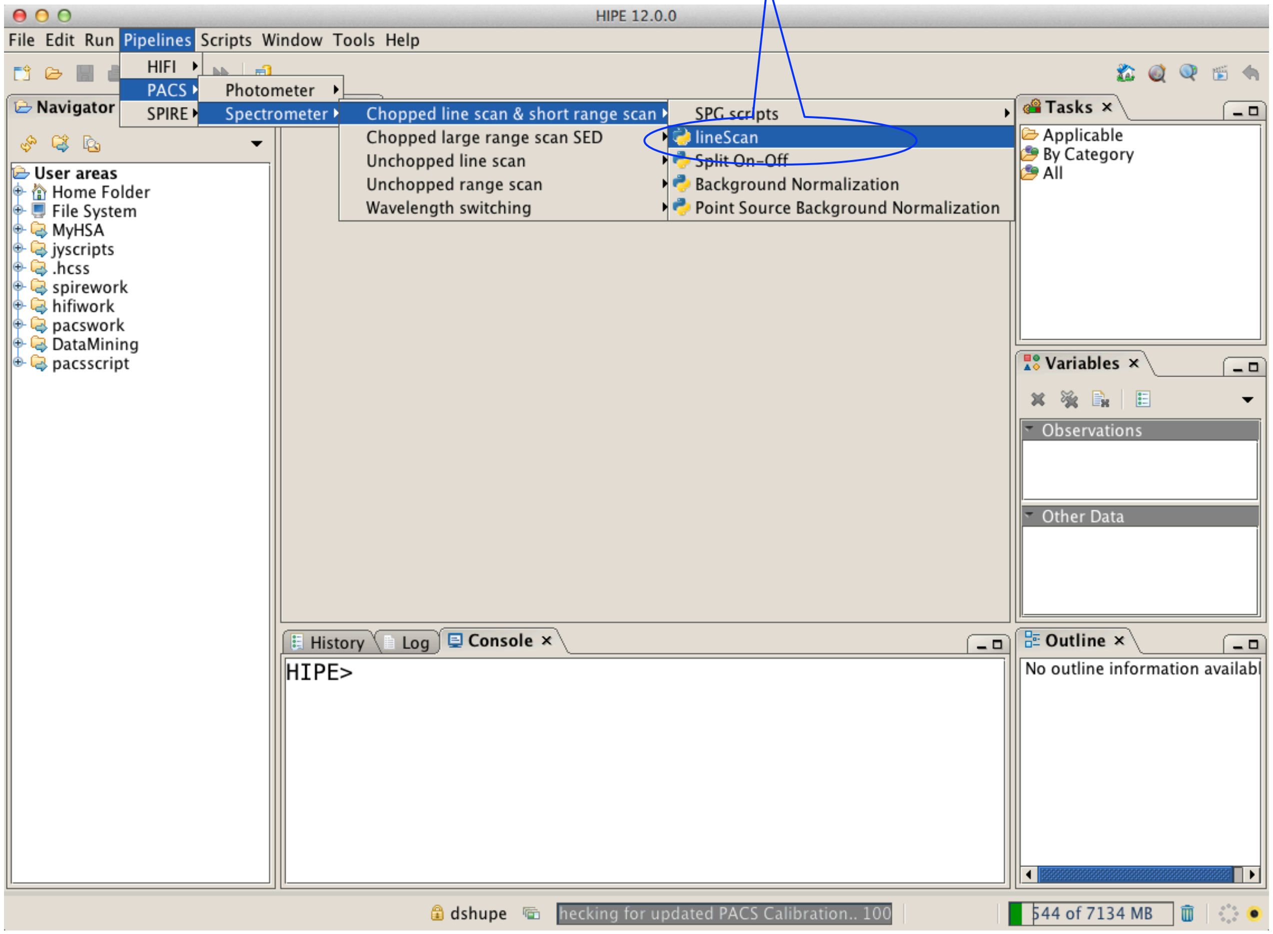
Step 2

Setup

Load pipeline script; load observation; check your data; and select the camera

Loading the script

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

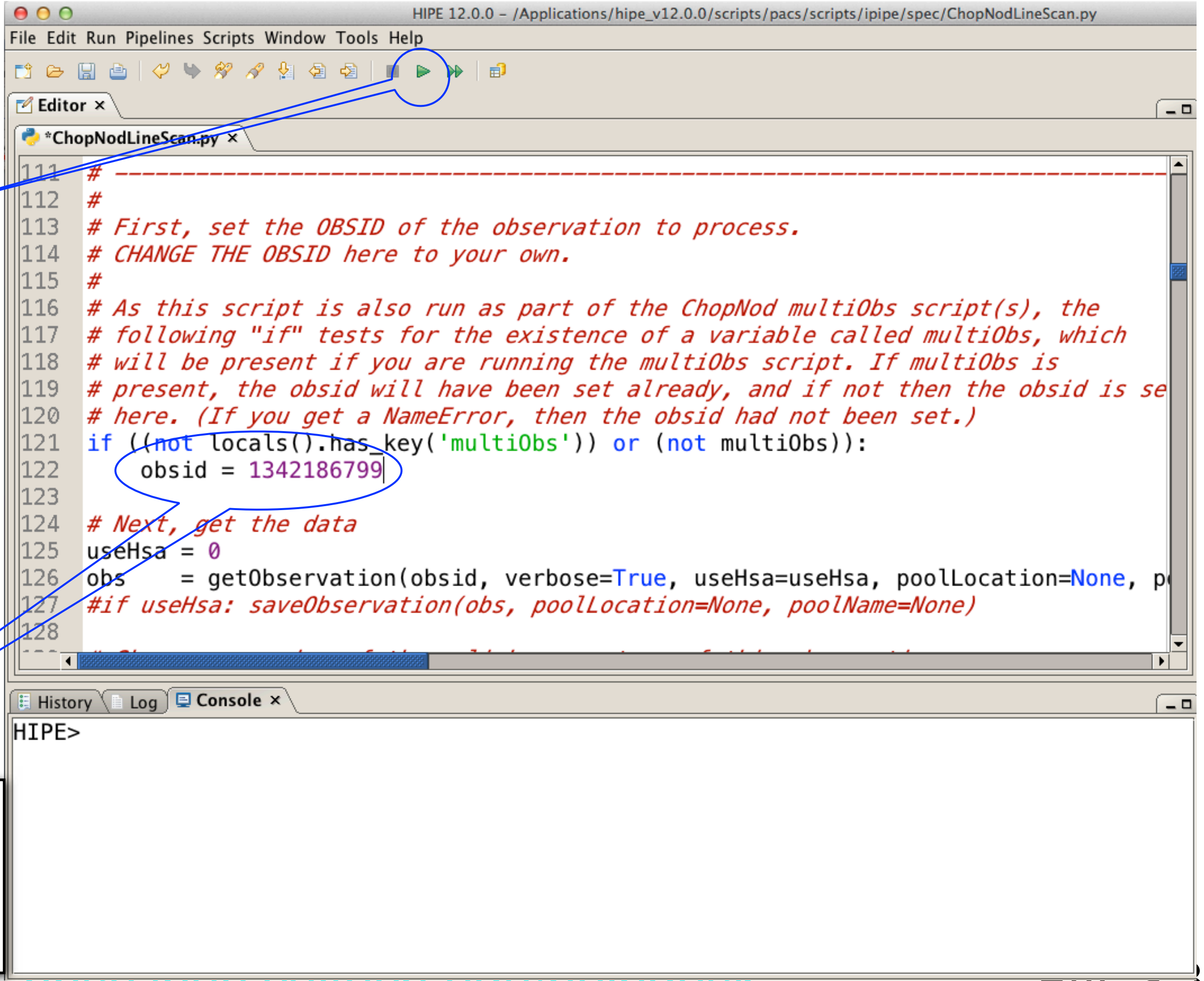


```
HIPE>
```

hecking for updated PACS Calibration.. 100

Loading the observation

Once the script is loaded, one simply steps through the lines to execute it. But first modify it for OBSID of the observation desired. Modify the obsid in the script and click through using the green arrow….


HIPE 12.0.0 – /Applications/hipe_v12.0.0/scripts/pacs/scripts/ipipe/spec/ChopNodLineScan.py
File Edit Run Pipelines Scripts Window Tools Help
Editor ×
*ChopNodLineScan.py ×
History
Log
Console ×

```
# ----------------------------------------------------------------------------------------
#
# First, set the OBSID of the observation to process.
# CHANGE THE OBSID here to your own.
#
# As this script is also run as part of the ChopNod multiObs script(s), the
# following "if" tests for the existence of a variable called multiObs, which
# will be present if you are running the multiObs script. If multiObs is
# present, the obsid will have been set already, and if not then the obsid is se
# here. (If you get a NameError, then the obsid had not been set.)
if ((not locals().has_key('multiObs')) or (not multiObs)):
    obsid = 1342186799

# Next, get the data
useHsa = 0
obs     = getObservation(obsid, verbose=True, useHsa=useHsa, poolLocation=None, p
#if useHsa: saveObservation(obs, poolLocation=None, poolName=None)
```

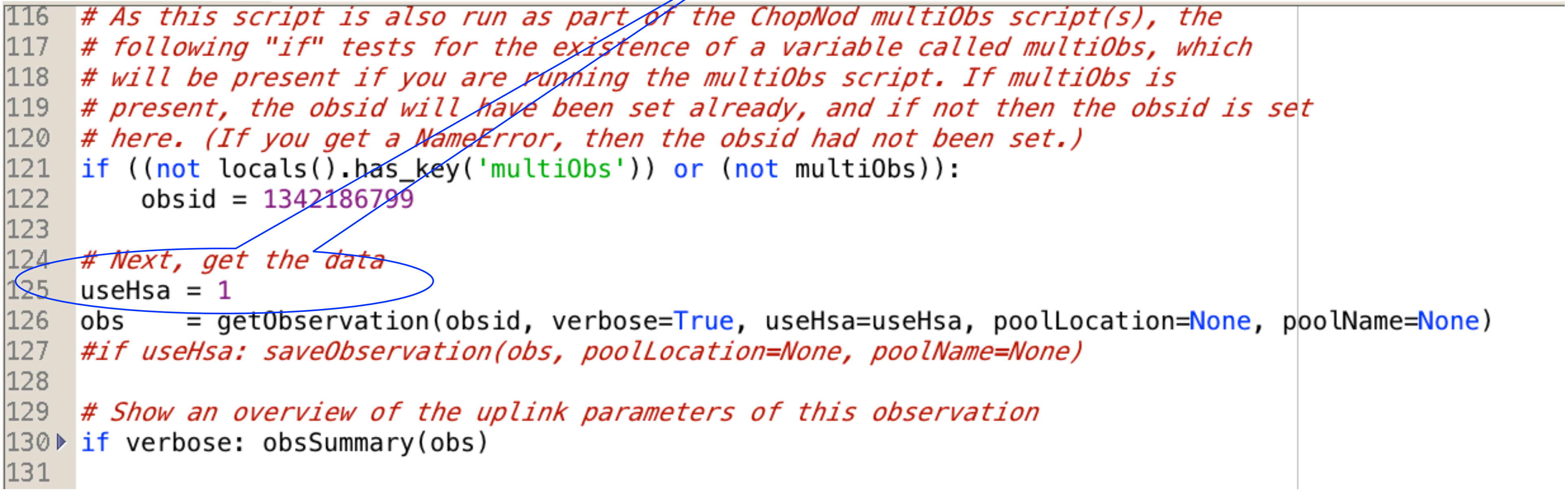
```
HIPE>
```

Hit the green arrow to step through the entire script

Modify this line. Here we set obsid to 1342186799.

Loading the observation

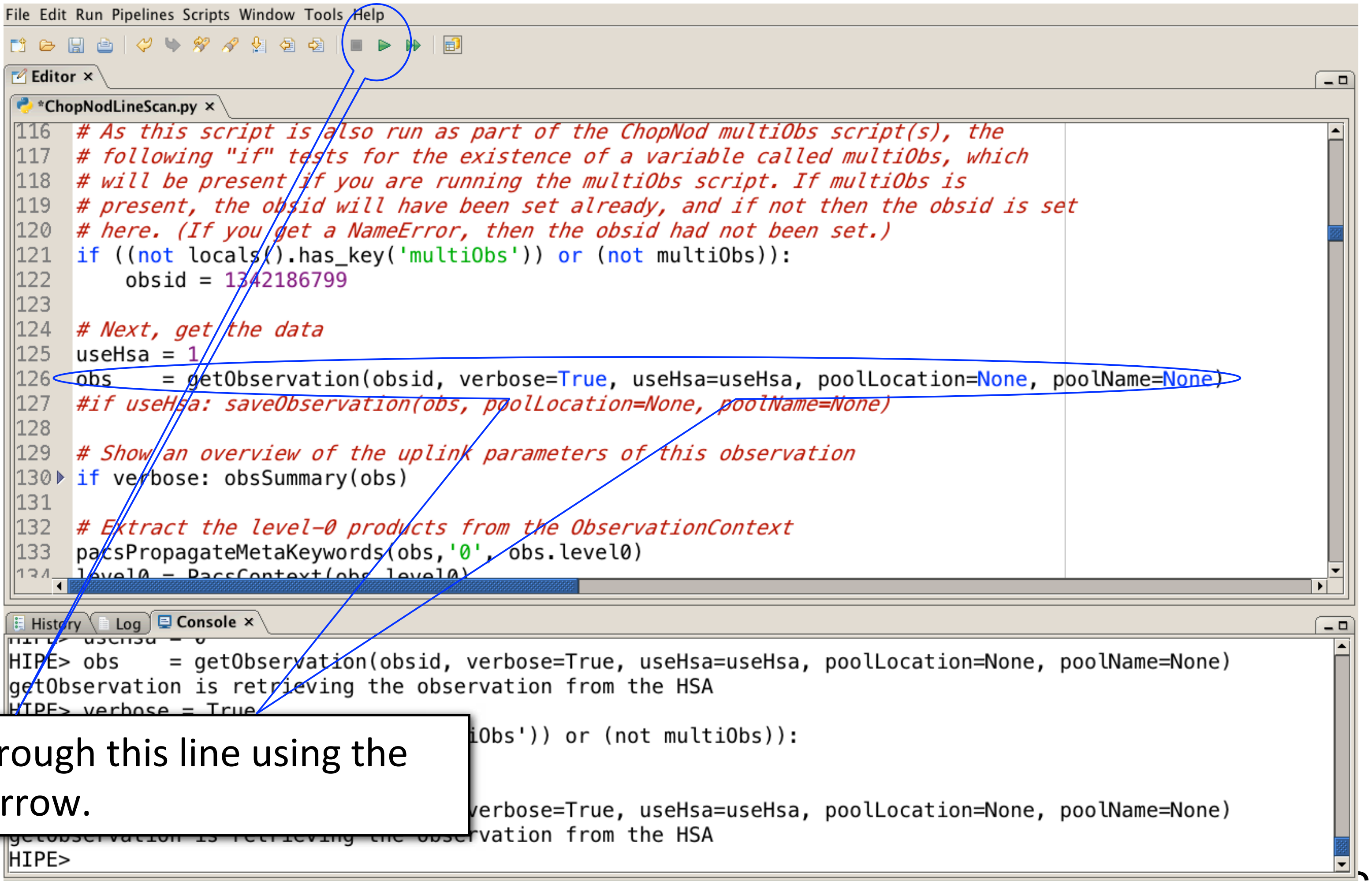
If the data is not stored as a local pool, you may need to tell getObservation to acquire the data from HSA. In this case edit the line to useHsa=1



```
# As this script is also run as part of the ChopNod multiObs script(s), the
# following "if" tests for the existence of a variable called multiObs, which
# will be present if you are running the multiObs script. If multiObs is
# present, the obsid will have been set already, and if not then the obsid is set
# here. (If you get a NameError, then the obsid had not been set.)
if ((not locals().has_key('multiObs')) or (not multiObs)):
    obsid = 1342186799

# Next, get the data
useHsa = 1
obs     = getObservation(obsid, verbose=True, useHsa=useHsa, poolLocation=None, poolName=None)
#if useHsa: saveObservation(obs, poolLocation=None, poolName=None)

# Show an overview of the uplink parameters of this observation
if verbose: obsSummary(obs)
```

Loading the observation

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

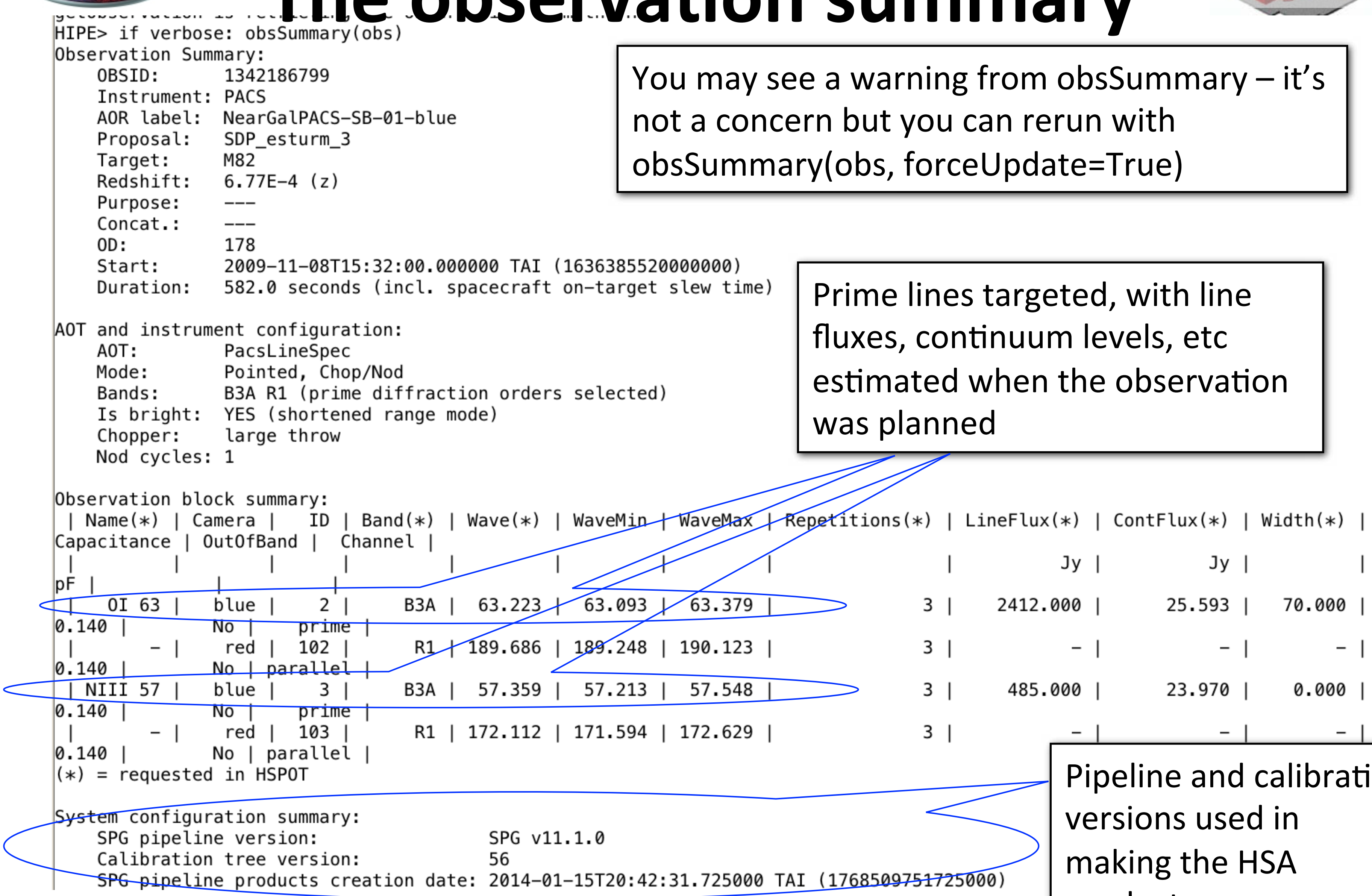


```
# As this script is also run as part of the ChopNod multiObs script(s), the
# following "if" tests for the existence of a variable called multiObs, which
# will be present if you are running the multiObs script. If multiObs is
# present, the obsid will have been set already, and if not then the obsid is set
# here. (If you get a NameError, then the obsid had not been set.)
if ((not locals().has_key('multiObs')) or (not multiObs)):
    obsid = 1342186799

# Next, get the data
useHsa = 1
obs     = getObservation(obsid, verbose=True, useHsa=useHsa, poolLocation=None, poolName=None)
#if useHsa: saveObservation(obs, poolLocation=None, poolName=None)

# Show an overview of the uplink parameters of this observation
if verbose: obsSummary(obs)

# Extract the level-0 products from the ObservationContext
pacsPropagateMetaKeywords(obs,'0', obs.level0)
level0 = PacsContext(obs.level0)
```

```
HIPE> obs     = getObservation(obsid, verbose=True, useHsa=useHsa, poolLocation=None, poolName=None)
getObservation is retrieving the observation from the HSA
HIPE> verbose = True
                                          iObs')) or (not multiObs)):

                                          verbose=True, useHsa=useHsa, poolLocation=None, poolName=None)
                                          vation from the HSA
HIPE>
```

Click through this line using the green arrow.

The observation summary



```
HIPE> if verbose: obsSummary(obs)
Observation Summary:
    OBSID:       1342186799
    Instrument: PACS
    AOR label:   NearGalPACS-SB-01-blue
    Proposal:    SDP_esturm_3
    Target:      M82
    Redshift:    6.77E-4 (z)
    Purpose:     ---
    Concat.:     ---
    OD:          178
    Start:       2009-11-08T15:32:00.000000 TAI (1636385520000000)
    Duration:    582.0 seconds (incl. spacecraft on-target slew time)

AOT and instrument configuration:
    AOT:         PacsLineSpec
    Mode:        Pointed, Chop/Nod
    Bands:       B3A R1 (prime diffraction orders selected)
    Is bright:   YES (shortened range mode)
    Chopper:     large throw
    Nod cycles: 1

Observation block summary:
 | Name(*) | Camera |   ID | Band(*) | Wave(*) | WaveMin | WaveMax | Repetitions(*) | LineFlux(*) | ContFlux(*) | Width(*) |
Capacitance | OutOfBand |  Channel |
 |         |        |      |         |         |         |         |                |          Jy |          Jy |          |
pF |         |        |
 |   OI 63 |   blue |    2 |     B3A |  63.223 |  63.093 |  63.379 |              3 |    2412.000 |      25.593 |   70.000 |
0.140 |         No |    prime |
 |       - |    red |  102 |      R1 | 189.686 | 189.248 | 190.123 |              3 |           - |           - |        - |
0.140 |         No | parallel |
 | NIII 57 |   blue |    3 |     B3A |  57.359 |  57.213 |  57.548 |              3 |     485.000 |      23.970 |    0.000 |
0.140 |         No |    prime |
 |       - |    red |  103 |      R1 | 172.112 | 171.594 | 172.629 |              3 |           - |           - |        - |
0.140 |         No | parallel |
(*) = requested in HSPOT

System configuration summary:
    SPG pipeline version:              SPG v11.1.0
    Calibration tree version:          56
    SPG pipeline products creation date: 2014-01-15T20:42:31.725000 TAI (1768509751725000)
```

You may see a warning from obsSummary – it's not a concern but you can rerun with obsSummary(obs, forceUpdate=True)

Prime lines targeted, with line fluxes, continuum levels, etc estimated when the observation was planned

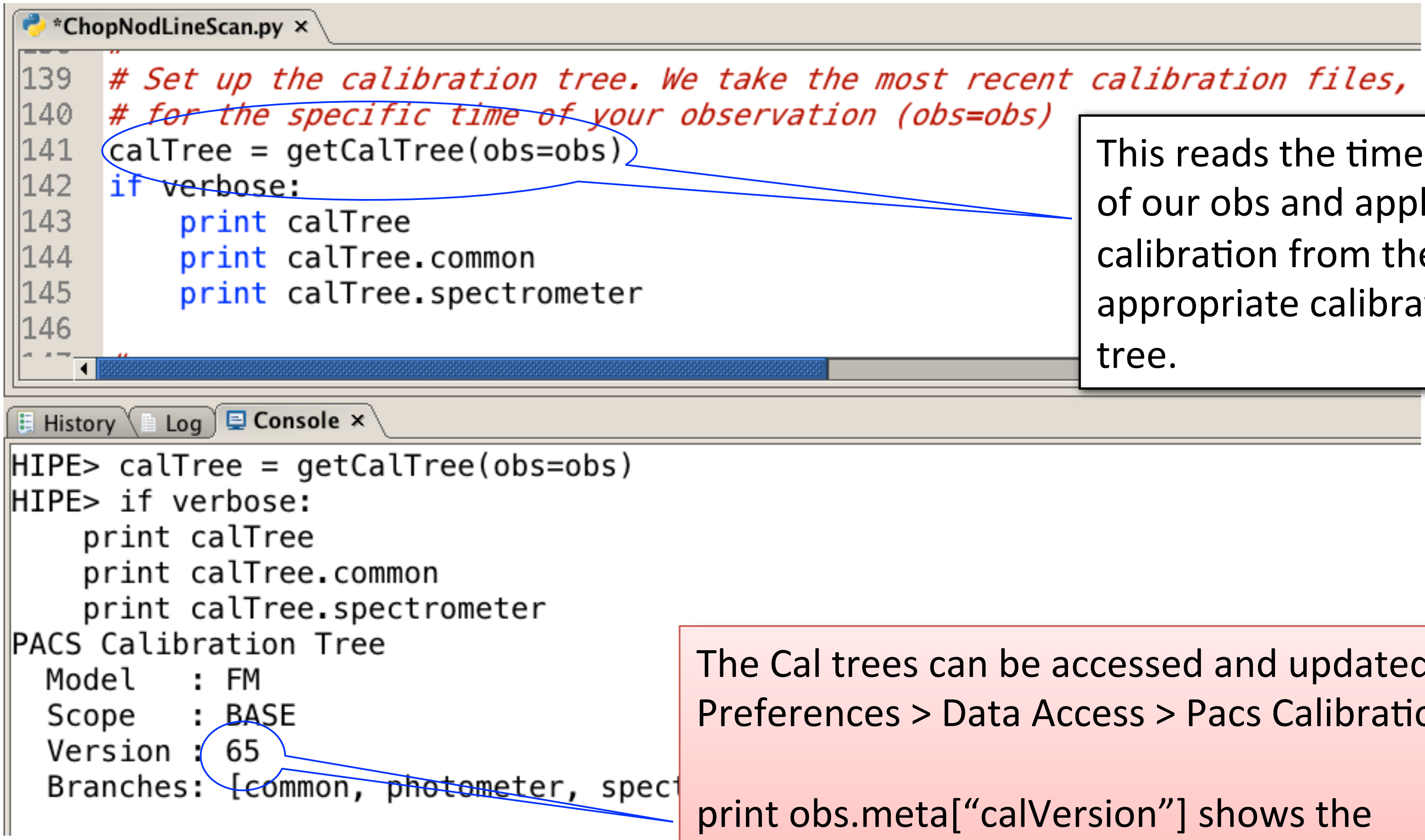
Pipeline and calibration versions used in making the HSA products

nhsc.ipac.caltech.edu/helpdesk

Retrieving the calibration tree

NHSC PACS Web Tutorial

Then, the calibration tree is loaded.



```
# Set up the calibration tree. We take the most recent calibration files,
# for the specific time of your observation (obs=obs)
calTree = getCalTree(obs=obs)
if verbose:
    print calTree
    print calTree.common
    print calTree.spectrometer
```

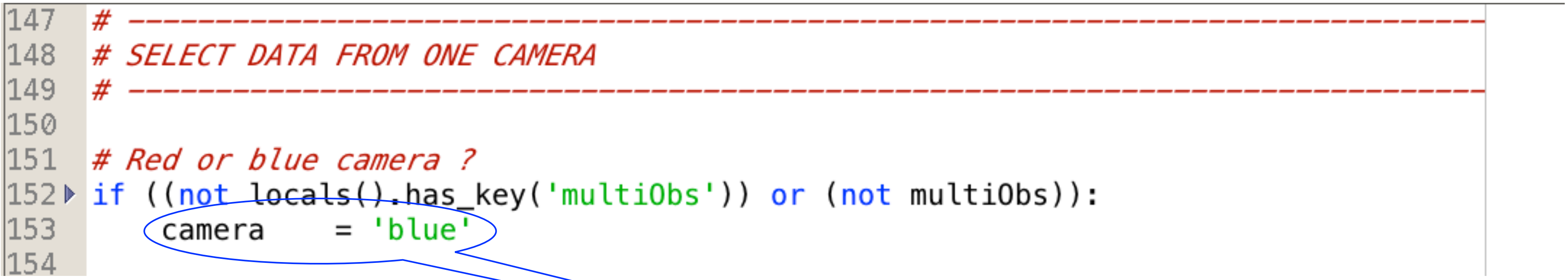
```
HIPE> calTree = getCalTree(obs=obs)
HIPE> if verbose:
    print calTree
    print calTree.common
    print calTree.spectrometer
PACS Calibration Tree
  Model    : FM
  Scope    : BASE
  Version  : 65
  Branches: [common, photometer, spect
```

This reads the time stamp of our obs and applies the calibration from the appropriate calibration tree.

The Cal trees can be accessed and updated from Preferences > Data Access > Pacs Calibration.

print obs.meta["calVersion"] shows the calibration used in current observation.

Setting the camera

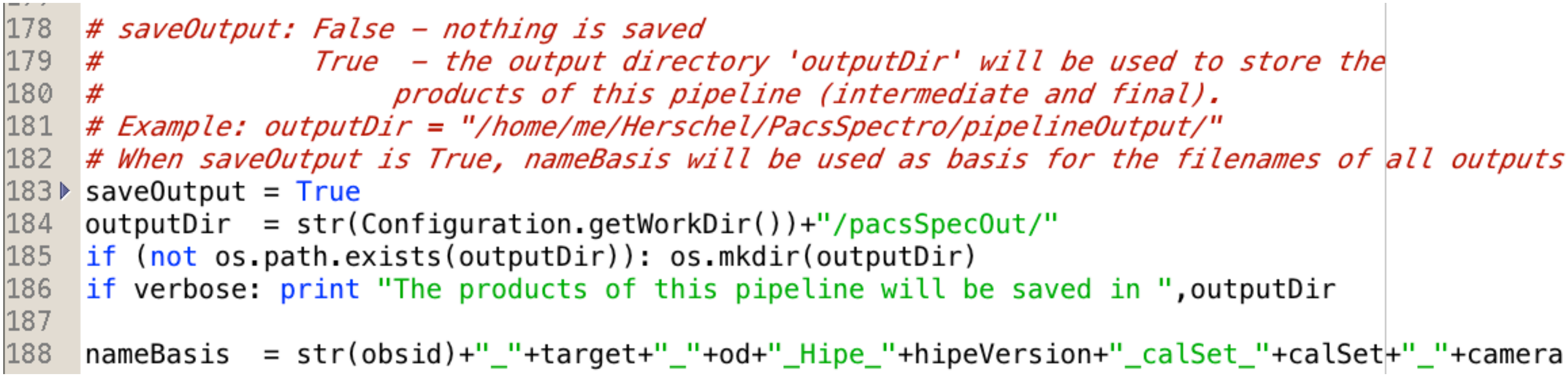


```
# ----------------------------------------------------------------------
# SELECT DATA FROM ONE CAMERA
# ----------------------------------------------------------------------

# Red or blue camera ?
if ((not locals().has_key('multiObs')) or (not multiObs)):
    camera     = 'blue'

```

We select camera = 'blue'

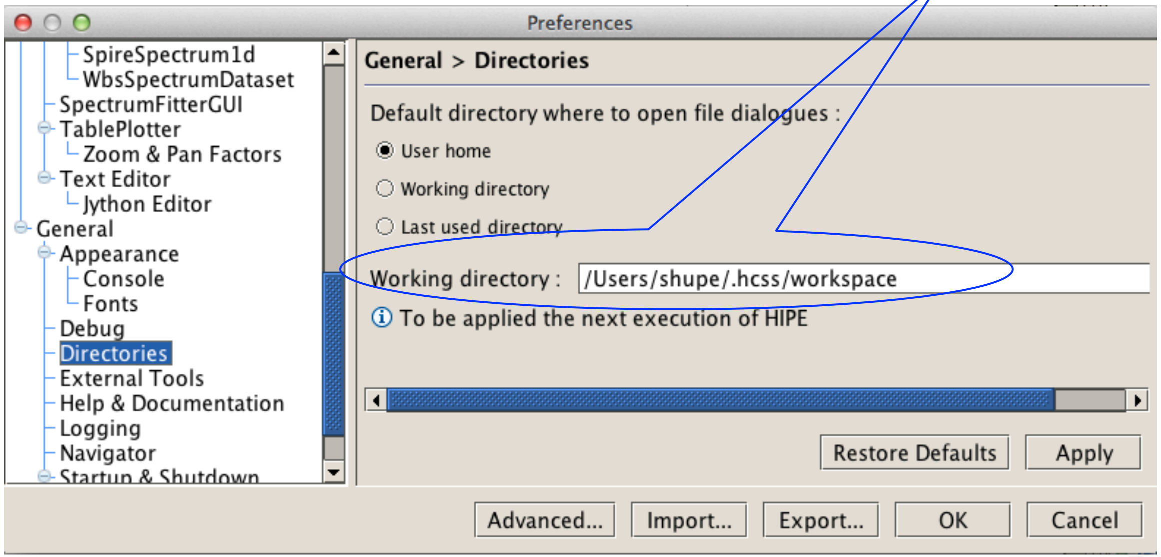
After selecting the camera, we can check what camera we selected by simply printing: “print camera”

Setting the output directory



```
# saveOutput: False - nothing is saved
#             True  - the output directory 'outputDir' will be used to store the
#                     products of this pipeline (intermediate and final).
# Example: outputDir = "/home/me/Herschel/PacsSpectro/pipelineOutput/"
# When saveOutput is True, nameBasis will be used as basis for the filenames of all outputs
saveOutput = True
outputDir  = str(Configuration.getWorkDir())+"/pacsSpecOut/"
if (not os.path.exists(outputDir)): os.mkdir(outputDir)
if verbose: print "The products of this pipeline will be saved in ",outputDir

nameBasis  = str(obsid)+"_"+target+"_"+od+"_Hipe_"+hipeVersion+"_calSet_"+calSet+"_"+camera
```

By default, the script will save intermediate and final products in your HIPE working directory. You can change the HIPE working directory using Edit -> Preferences -> Directories.



General > Directories

Default directory where to open file dialogues :

● User home

○ Working directory

○ Last used directory

Working directory : /Users/shupe/.hcss/workspace

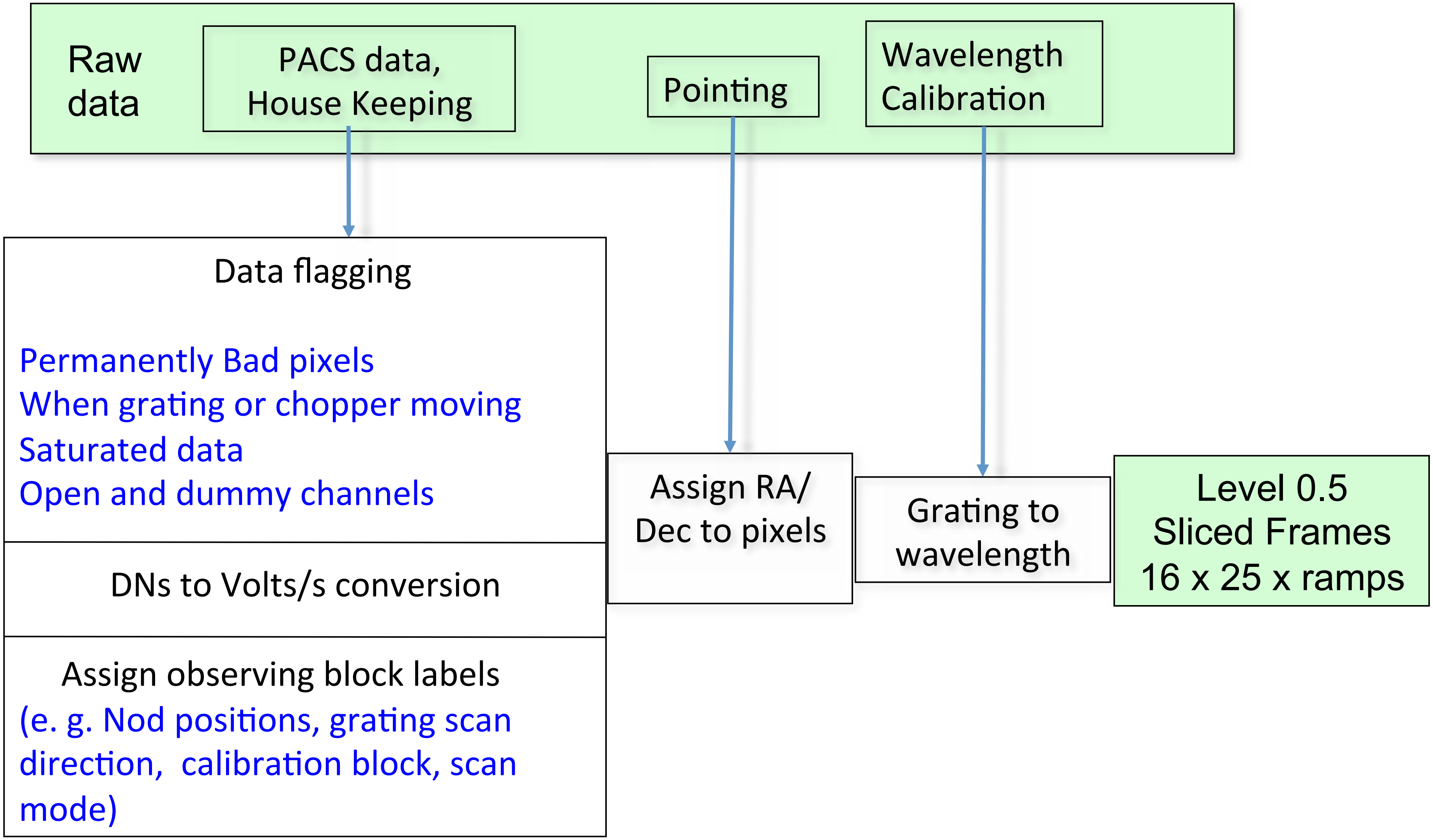
ⓘ To be applied the next execution of HIPE

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

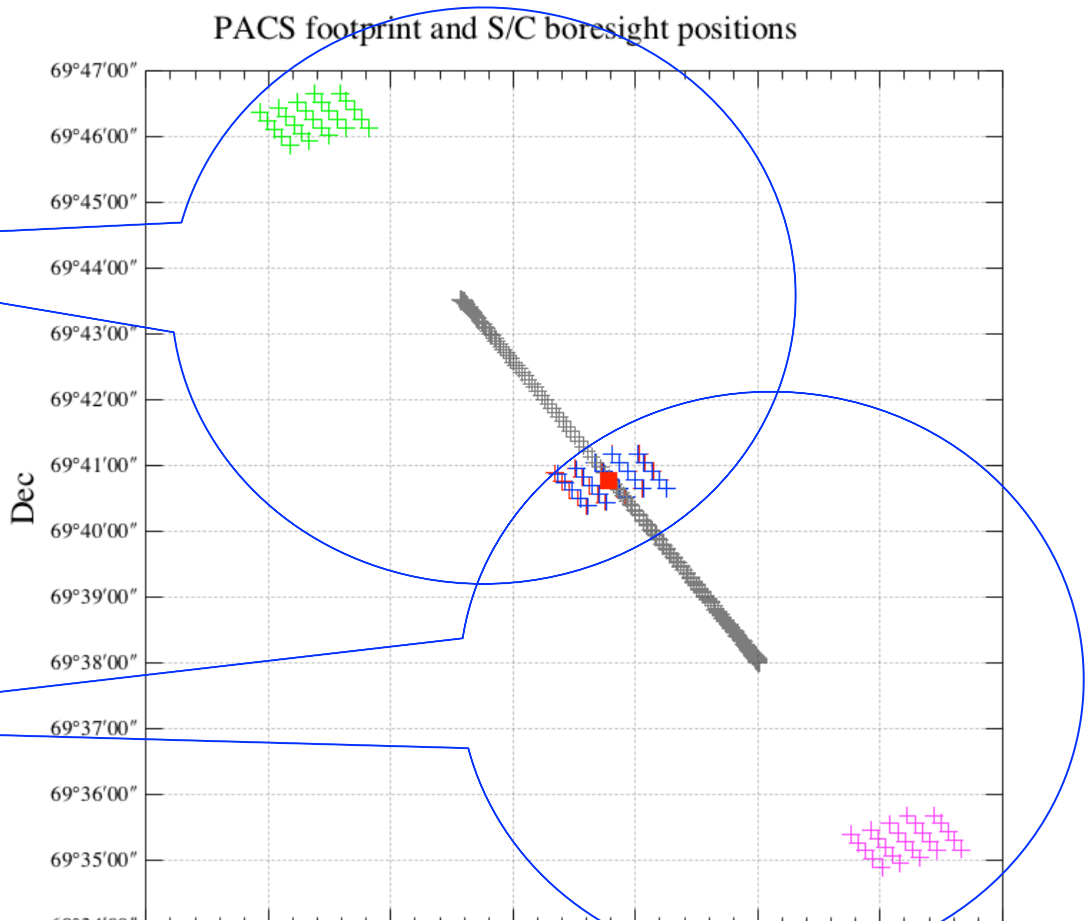
Assign observing block labels
(e. g. Nod positions, grating scan direction, calibration block, scan mode)

Assign RA/Dec to pixels

Grating to wavelength

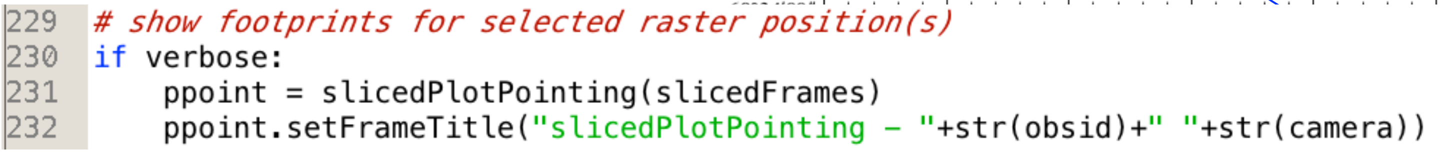
Level 0.5
Sliced Frames
16 x 25 x ramps

Plot: footprint



Nod A

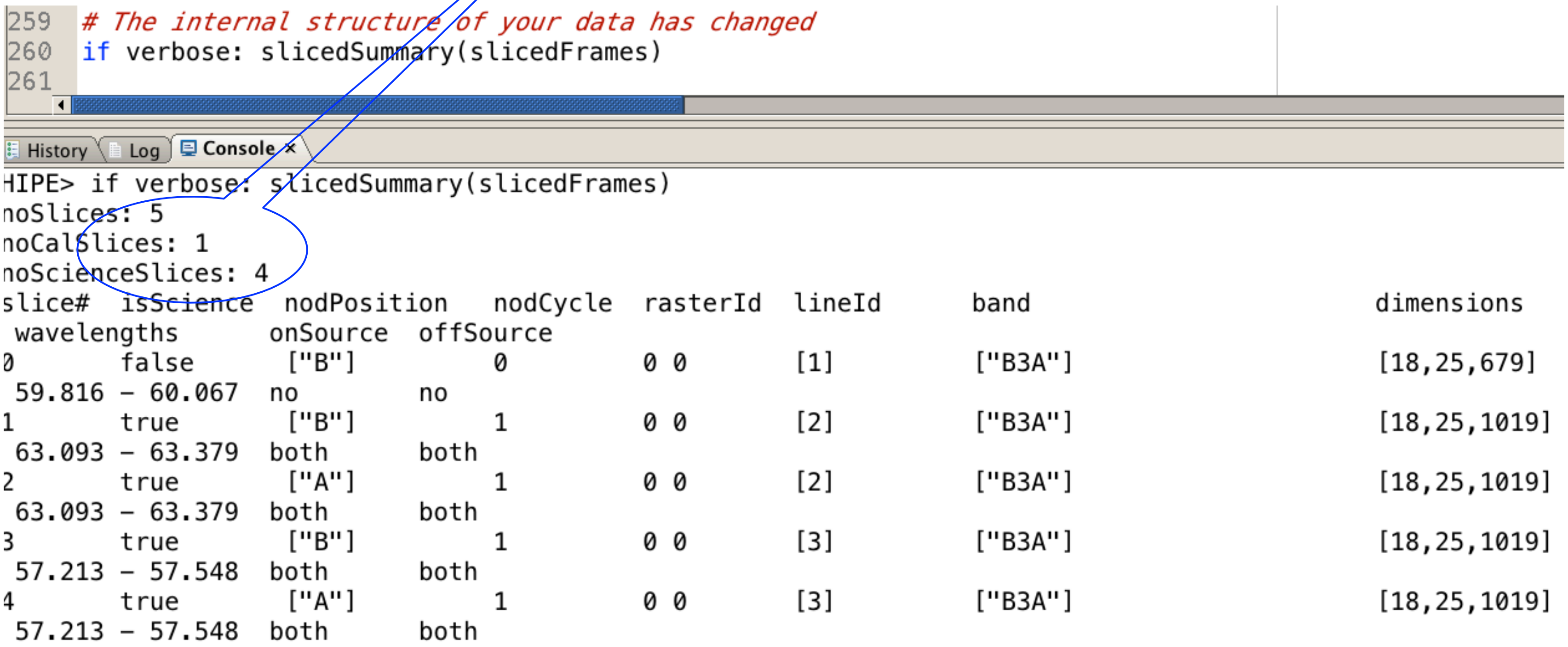
Nod B



```
# show footprints for selected raster position(s)
if verbose:
    ppoint = slicedPlotPointing(slicedFrames)
    ppoint.setFrameTitle("slicedPlotPointing – "+str(obsid)+" "+str(camera))
```

Slicing into nods

The slicing of the data is performed according to rules made explicit in the pipeline. In our example, two lines are observed in two nodding positions. So, we expect 4 slices plus an initial slice containing the calibration block.



```
# The internal structure of your data has changed
if verbose: slicedSummary(slicedFrames)
```

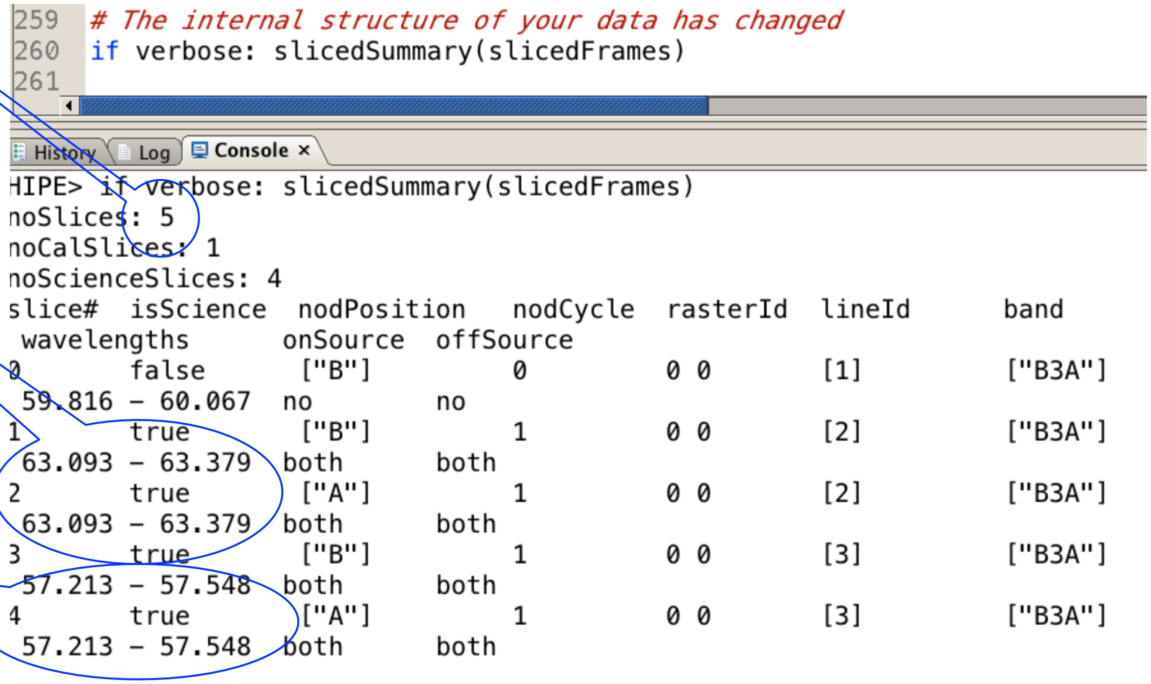
```
HIPE> if verbose: slicedSummary(slicedFrames)
noSlices: 5
noCalSlices: 1
noScienceSlices: 4
slice#  isScience  nodPosition  nodCycle  rasterId  lineId     band                       dimensions
 wavelengths       onSource   offSource
0       false       ["B"]        0        0 0       [1]        ["B3A"]                    [18,25,679]
 59.816 - 60.067   no         no
1       true        ["B"]        1        0 0       [2]        ["B3A"]                    [18,25,1019]
 63.093 - 63.379   both       both
2       true        ["A"]        1        0 0       [2]        ["B3A"]                    [18,25,1019]
 63.093 - 63.379   both       both
3       true        ["B"]        1        0 0       [3]        ["B3A"]                    [18,25,1019]
 57.213 - 57.548   both       both
4       true        ["A"]        1        0 0       [3]        ["B3A"]                    [18,25,1019]
 57.213 - 57.548   both       both
```

Check: after slicing

5 slices !

Line 1 – B & A nodes

Line 2 – B & A nodes

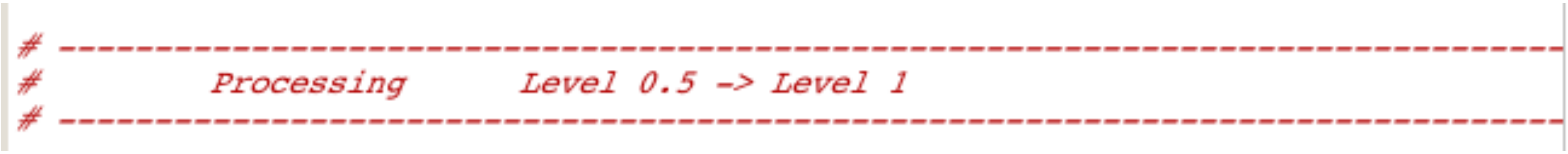


```
# The internal structure of your data has changed
if verbose: slicedSummary(slicedFrames)
```

```
HIPE> if verbose: slicedSummary(slicedFrames)
noSlices: 5
noCalSlices: 1
noScienceSlices: 4
slice#  isScience  nodPosition   nodCycle  rasterId  lineId      band
 wavelengths      onSource  offSource
0       false       ["B"]         0         0 0       [1]         ["B3A"]
 59.816 - 60.067  no        no
1       true        ["B"]         1         0 0       [2]         ["B3A"]
 63.093 - 63.379  both      both
2       true        ["A"]         1         0 0       [2]         ["B3A"]
 63.093 - 63.379  both      both
3       true        ["B"]         1         0 0       [3]         ["B3A"]
 57.213 - 57.548  both      both
4       true        ["A"]         1         0 0       [3]         ["B3A"]
 57.213 - 57.548  both      both
```

Continue …

With remaining Level 0 to 0.5 processing steps as outlined in slide 18. Step through with the green arrow.



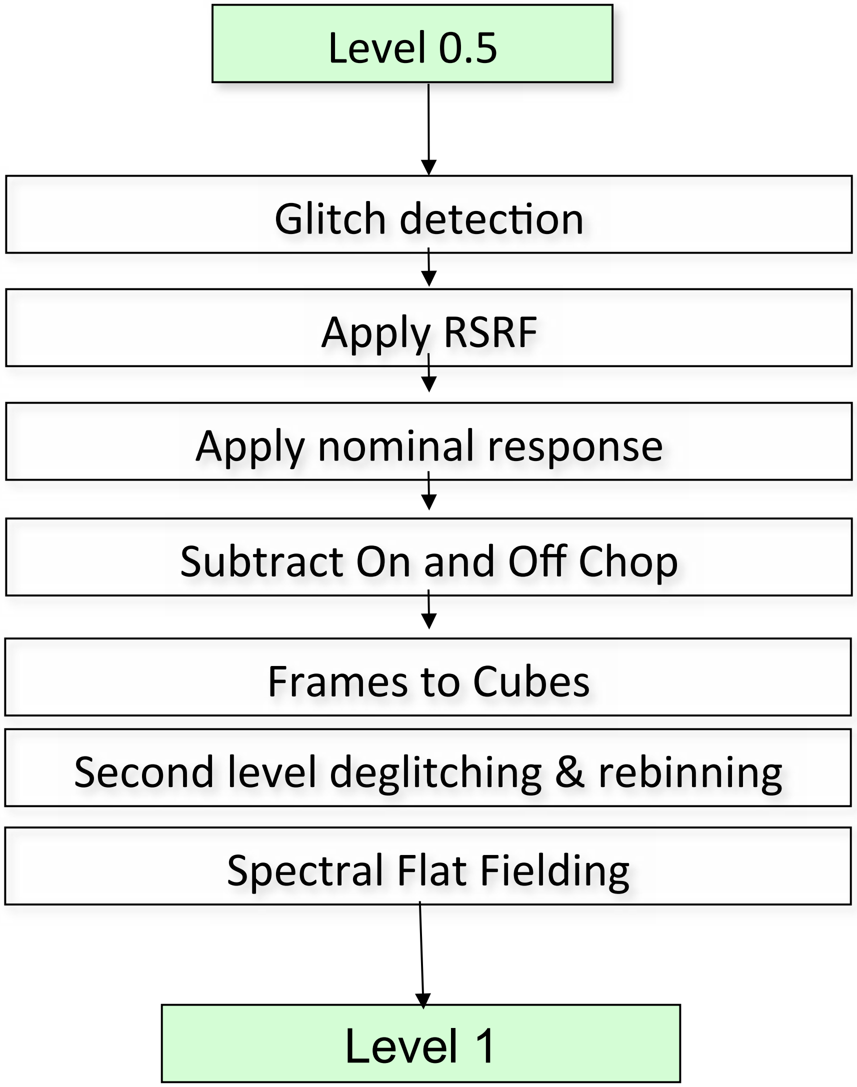
```
# ---------------------------------------------------------------------------
#         Processing        Level 0.5 -> Level 1
# ---------------------------------------------------------------------------
```

Step 4

Run the 0.5 → 1 pipeline

Glitch detection, chop differentiation, RSRF, flat

Level 0.5 → 1



Level 0.5

Glitch detection

Apply RSRF

Apply nominal response

Subtract On and Off Chop

Frames to Cubes

Second level deglitching & rebinning

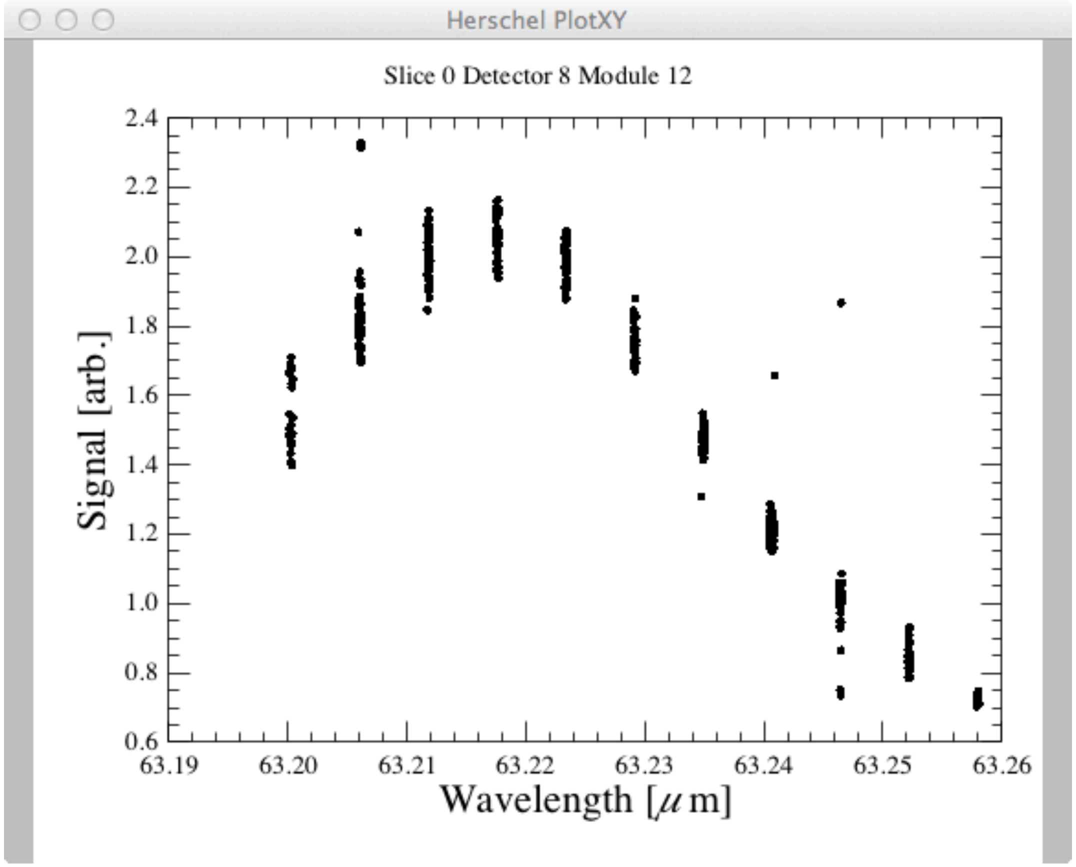
Spectral Flat Fielding

Level 1

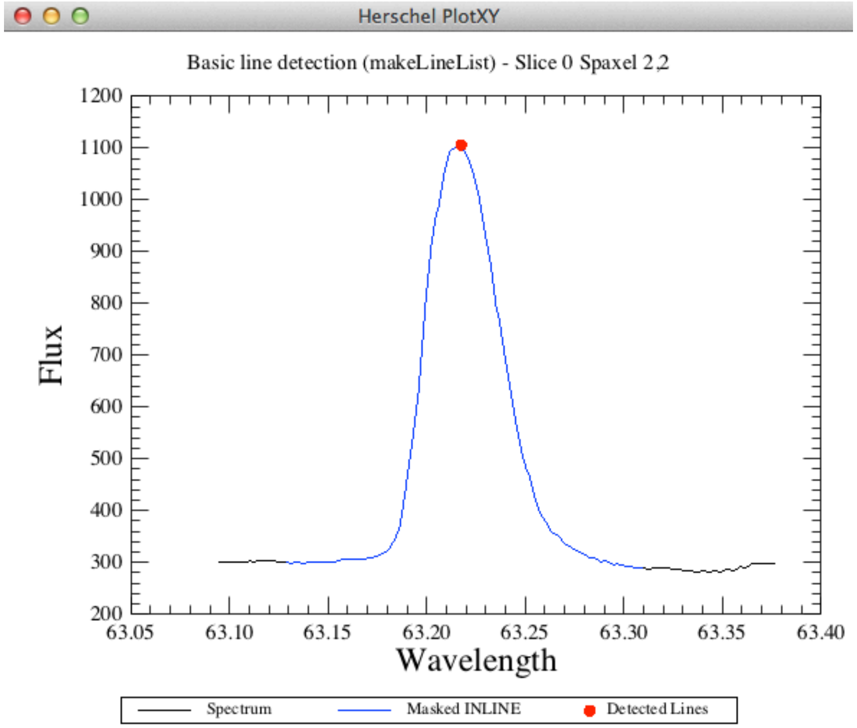
Signal after chop subtraction

Verbose=1 shows

The data are only on the ON position (OFF being subtracted)

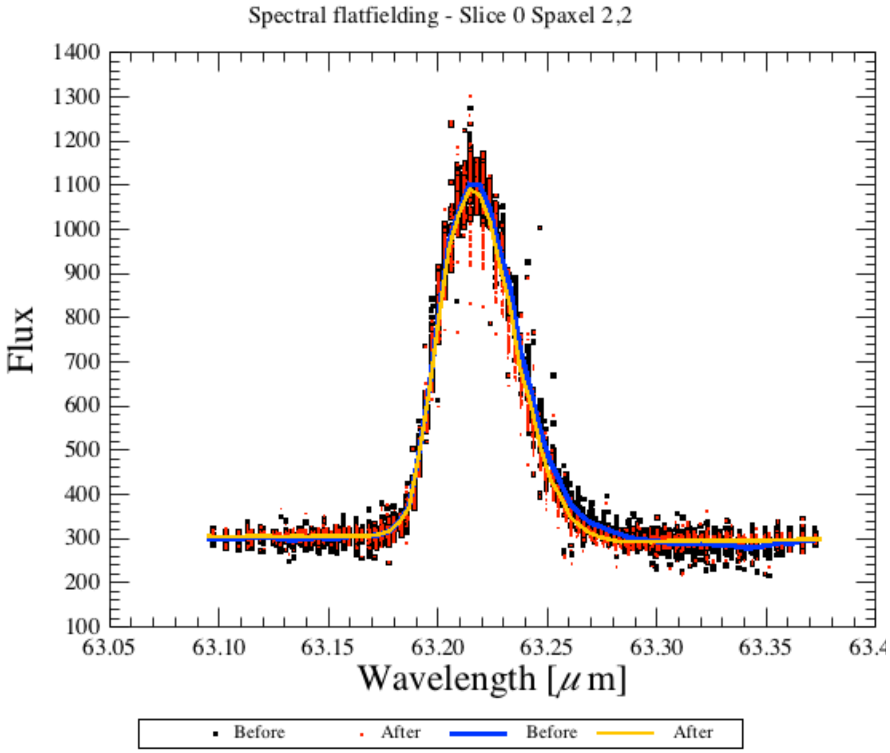


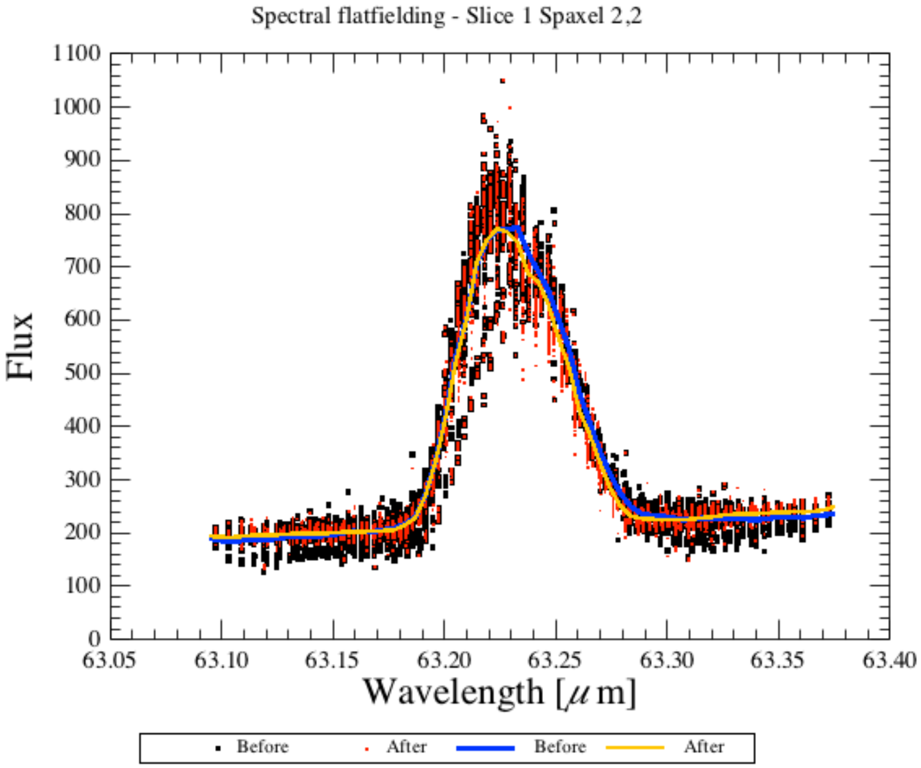
Check: Spectral FlatField



As a default, the code will search for lines in all the pixels and then mask them before computing the spectral flat field.
It is possible to give directly the list of lines to be masked via the parameter lineList = [63.227], for instance.
This user-specified lineList is usually needed *only* for absorption lines.

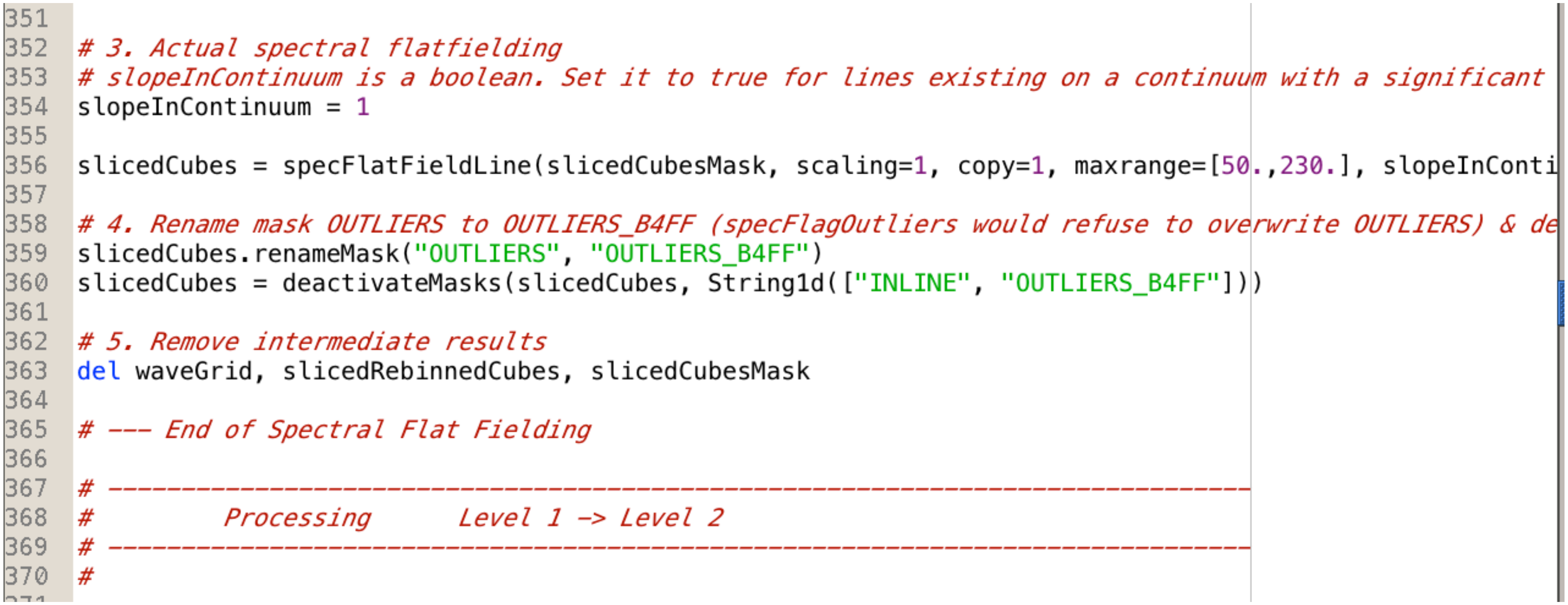
Check: Spectral FlatField





At this stage you will just want to check that the red "After" points have a tighter distribution (less scatter) than the black "Before" points.

You are ready to continue with PACS-302



```
# 3. Actual spectral flatfielding
# slopeInContinuum is a boolean. Set it to true for lines existing on a continuum with a significant
slopeInContinuum = 1

slicedCubes = specFlatFieldLine(slicedCubesMask, scaling=1, copy=1, maxrange=[50.,230.], slopeInConti

# 4. Rename mask OUTLIERS to OUTLIERS_B4FF (specFlagOutliers would refuse to overwrite OUTLIERS) & de
slicedCubes.renameMask("OUTLIERS", "OUTLIERS_B4FF")
slicedCubes = deactivateMasks(slicedCubes, String1d(["INLINE", "OUTLIERS_B4FF"]))

# 5. Remove intermediate results
del waveGrid, slicedRebinnedCubes, slicedCubesMask

# --- End of Spectral Flat Fielding

# ------------------------------------------------------------------------------
#         Processing      Level 1 -> Level 2
# ------------------------------------------------------------------------------
#
```