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
Updated for HIPE 13.0.0 by Roberta Paladini, June 2015

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 OBSID

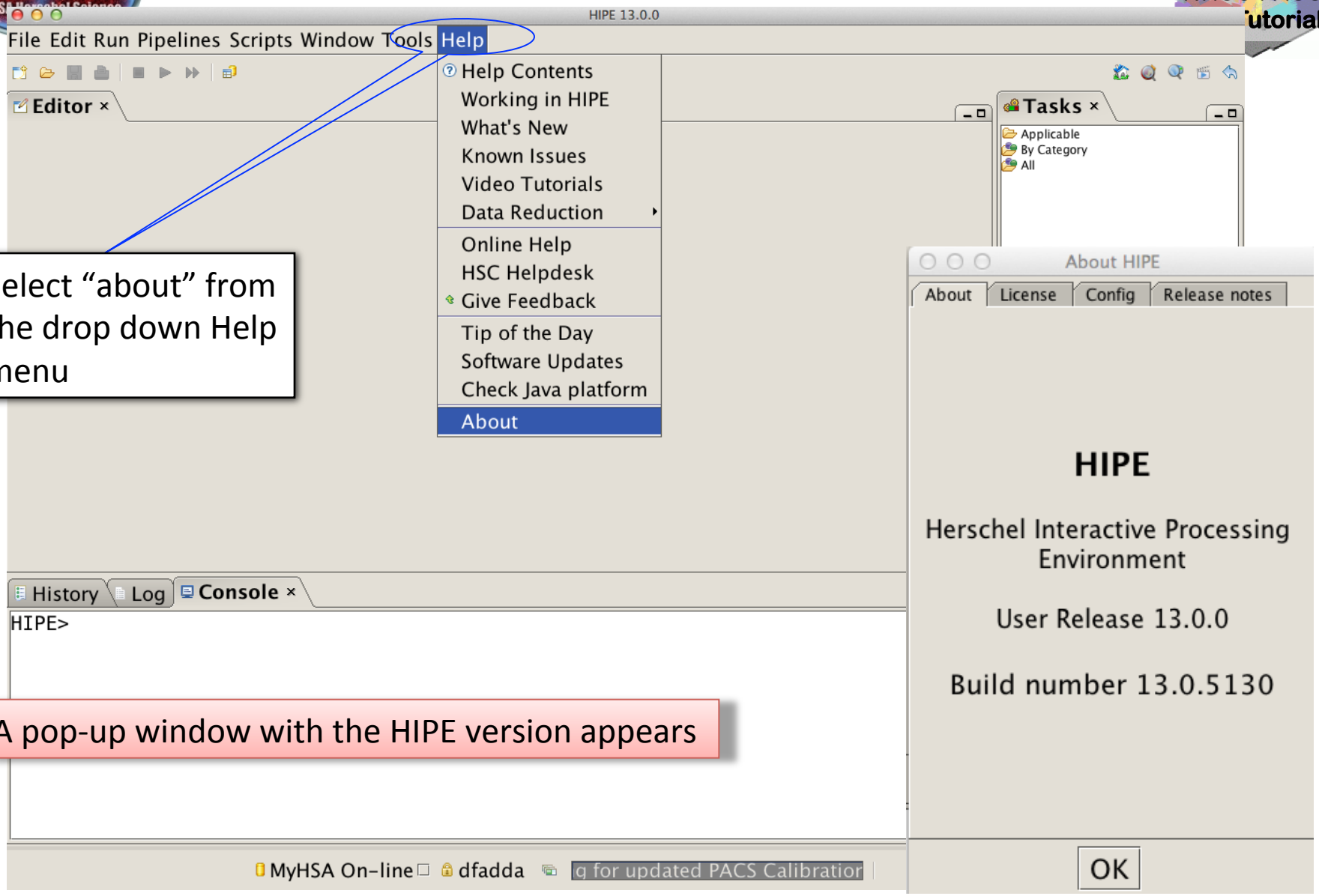
Step 3 Run the 0 → 0.5 pipeline

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 13.0.0,
also known as Build number 13.0.5130

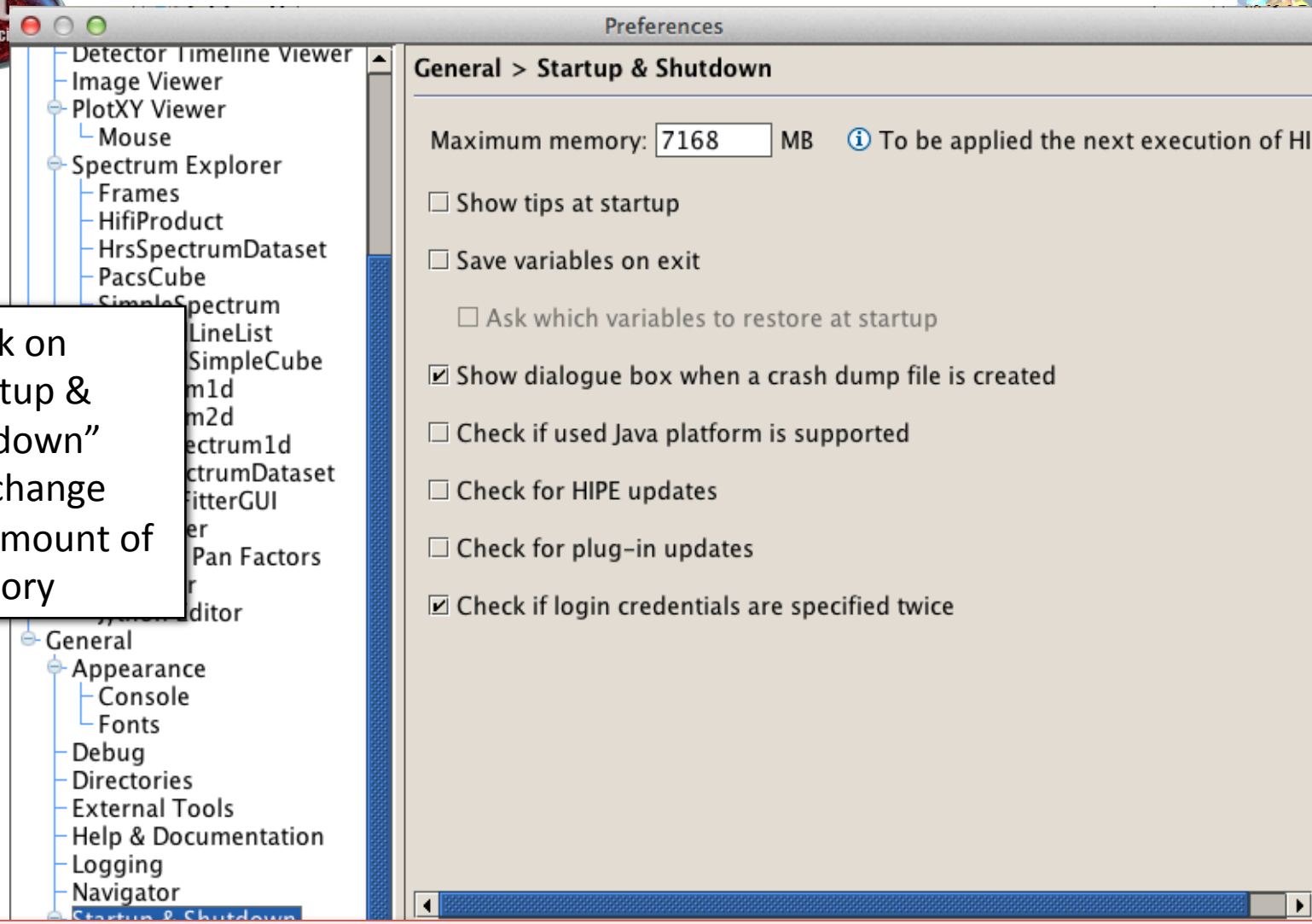


Select “about” from the drop down Help menu

A pop-up window with the HIPE version appears

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

N.b.:, Memory used and available



…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.

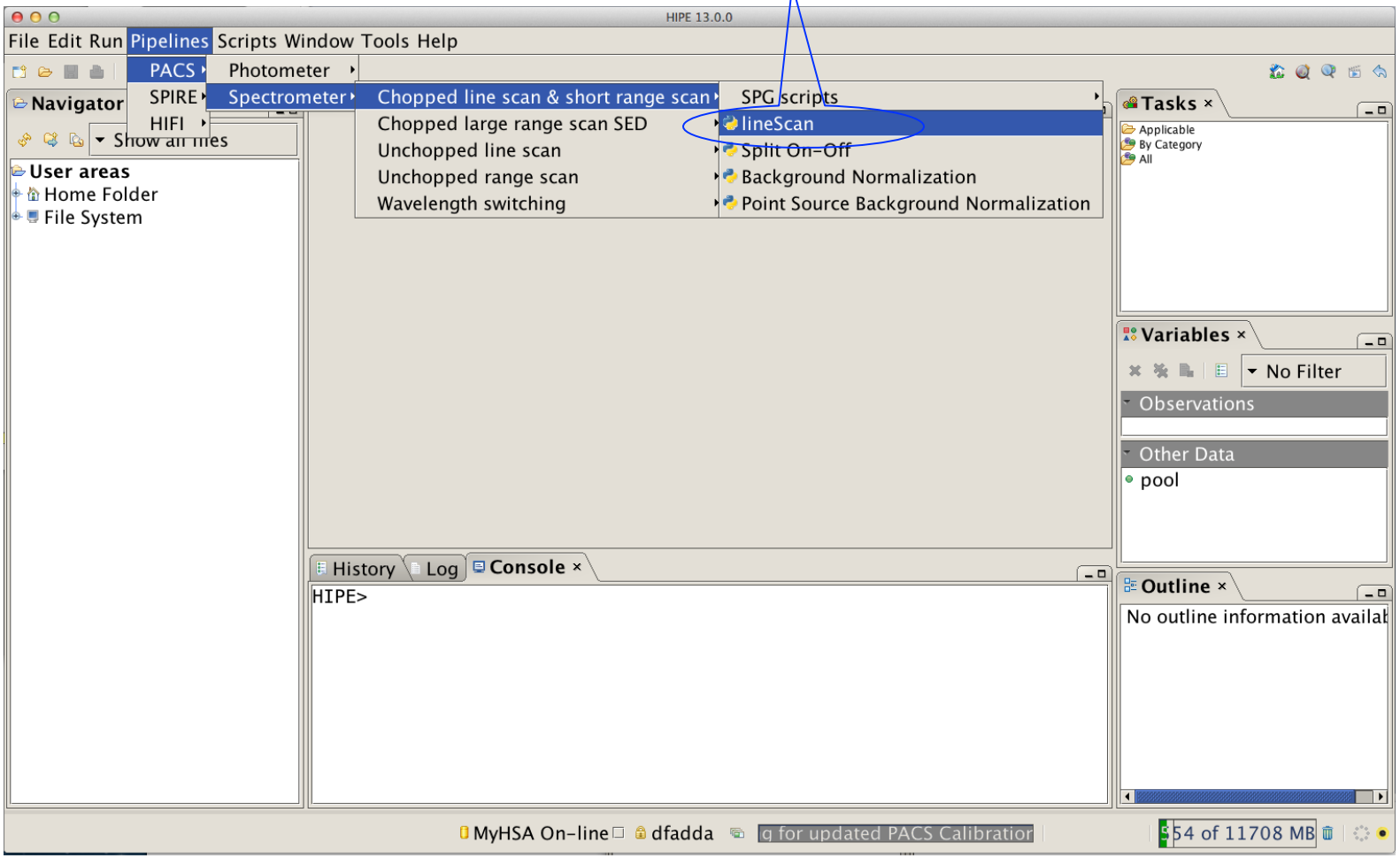
Step 2

Setup

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

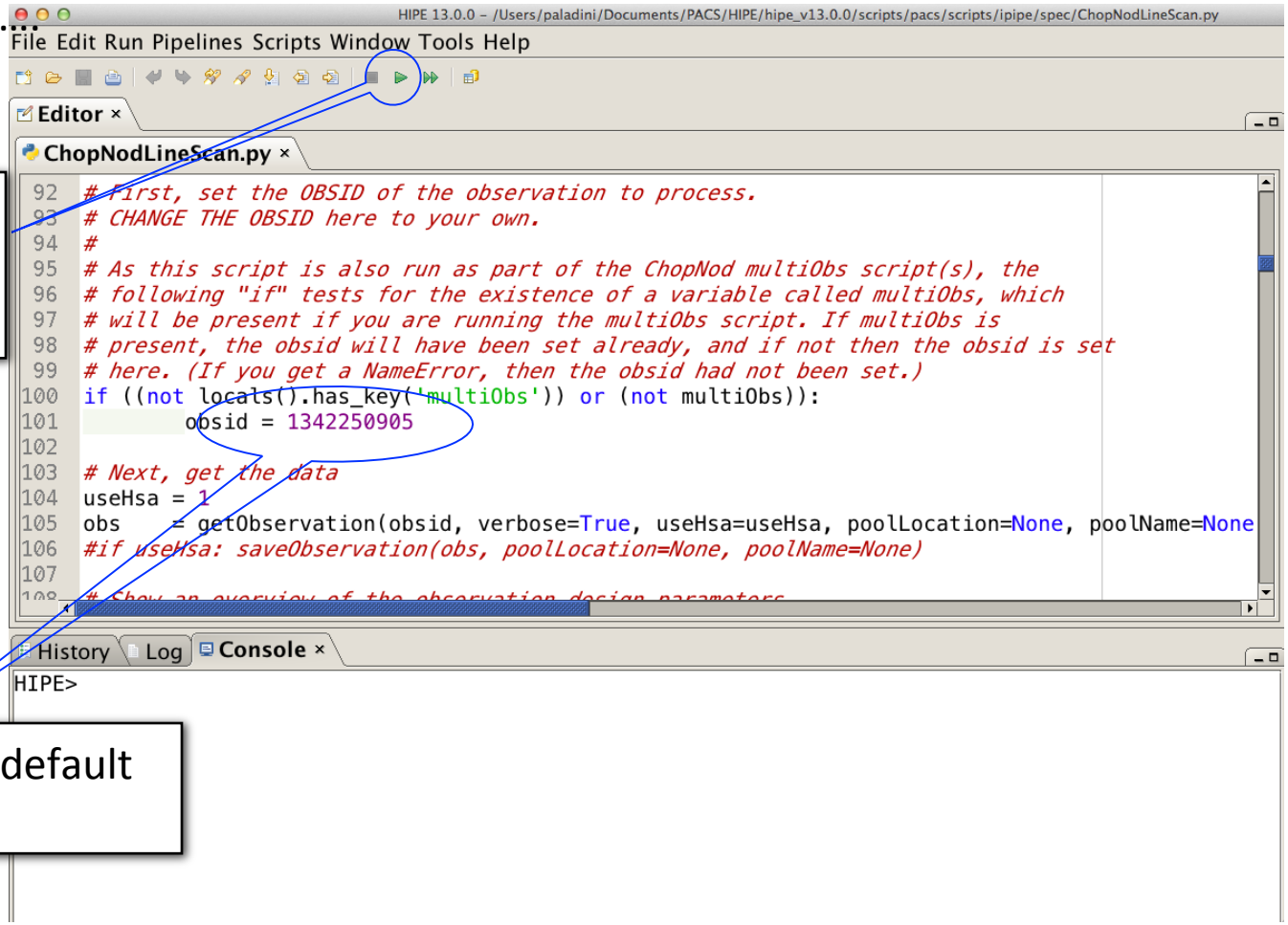
Loading the script

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



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...



Hit the green arrow to step through the entire script

Modify this line. The default is obsid 1342250905

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, make sure 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 = 1342250905

# 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 observation design parameters
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).

File Edit Run Pipelines Scripts Window Tools Help

Editor ×

ChopNodLineScan.py ×

```
94  #
95  # As this script is also run as part of the ChopNod multiObs script(s), the
96  # following "if" tests for the existence of a variable called multiObs, which
97  # will be present if you are running the multiObs script. If multiObs is
98  # present, the obsid will have been set already, and if not then the obsid is set
99  # here. (If you get a NameError, then the obsid had not been set.)
100 if ((not locals().has_key('multiObs')) or (not multiObs)):
101         obsid = 1342250905
102
103 # Next, get the data
104 useHsa = 1
105 obs     = getObservation(obsid, verbose=True, useHsa=useHsa, poolLocation=None, poolName=None)
106 #if useHsa: saveObservation(obs, poolLocation=None, poolName=None)
107
108 # Show an overview of the observation design parameters
109 if verbose: obsSummary(obs)
110
111 # Extract the level-0 products from the ObservationContext
112 pacsPropagateMetaKeywords(obs,'0', obs.level0)
```

History | Log | Console ×

```
HIPE> import os,sys
HIPE> verbose = True
HIPE> if ((not locals().has_key('multiObs')) or (not multiObs)):

                                    verbose=True, useHsa=useHsa, poolLocation=None, poolName=None)
                                    rvation from the HSA
```

Click through this line using the green arrow.

The observation summary

```
Observation Summary:
    OBSID:        1342250905
    Instrument:   PACS
    AOR label:    RedRectangle-OI
    Proposal:     OT1_vbujarra_4
    Target:       Red Rectangle
    Actual RA:    6h 19m 58.27s
    Actual Dec.:  -10° 38' 14.68''
    Redshift:     0.0 (rad. vel. km/s)
    Purpose:      ---
    Concat.:      ---
    OD:           1217
    Start:        2012-09-11T19:17:13.000000 TAI (1726082233000000)
    Duration:     1315.0 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:   NO (default range mode)
    Chopper:     medium throw
    Nod cycles:  3

Observation block summary:
 |     Name(*) | Camera |   ID | Band(*) |    Wave(*) |    WaveMin |    WaveMax | Repetitions(*) | ActualRep | Capacitance | OutOfBand |
Channel |
 |             |        |      |         | micrometer | micrometer | micrometer |                |           |          pF |           |
 |
 | O I 3P1-3P2 |   blue |    2 |     B3A |     63.180 |     62.936 |     63.441 |              1 |         1 |       0.140 |        No |
 prime |
 |           - |    red |  102 |      R1 |    189.543 |    188.776 |    190.310 |              1 |         1 |       0.140 |        No |
parallel |
(*) = requested in HSPOT

System configuration summary:
    SPG pipeline version:                SPG v13.0.0
    SPG pipeline products creation date: 2015-05-22T22:20:54.680000 TAI (1811024454680000)
    Mission configuration:               MC_H102ASTR_P70ASTR_S66ASTR_RP
    Processed to level:                  RED: 2.0 BLUE: 2.0
    Quality Control:                     PENDING
    Action:                              NONE
```

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

Prime lines targeted when the observation was planned

Pipeline version used in making the HSA products

Retrieving the calibration tree

Then, the calibration tree is loaded.

ChopNodLineScan.py

```
# 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
```

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

History | Log | Console

```
PACS Calibration Tree
  Model   : FM
  Scope   : BASE
  Version : 69
  Branches: [common, photometer, spect

PacsCalCommon Calibration Products:
  chopperAngle                    : FM
  chopperAngleRedundant           : FM
```

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"

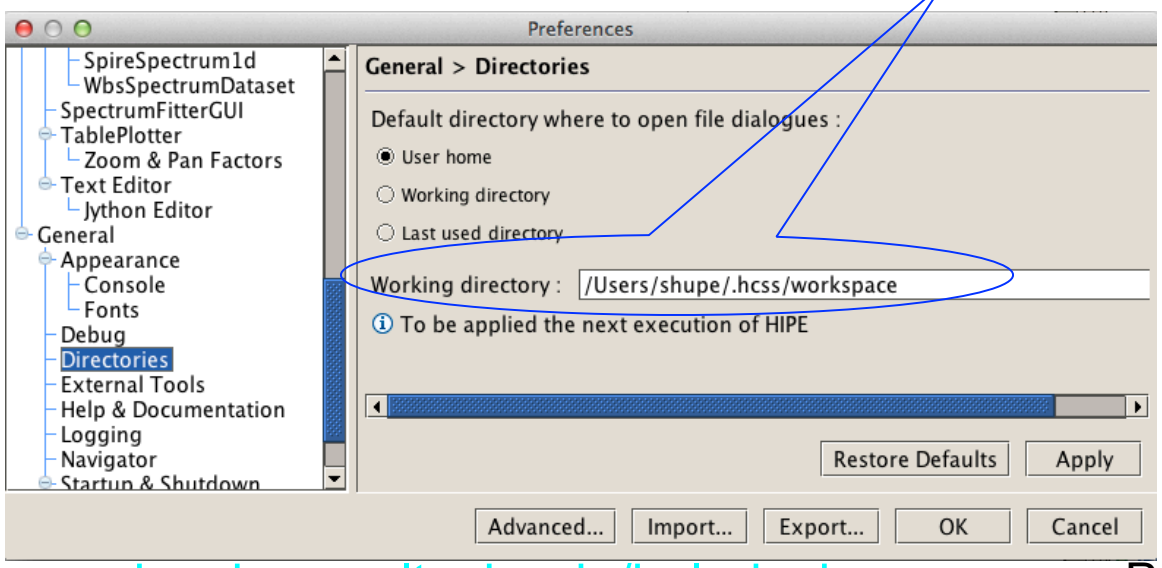
Setting the output directory

```
# saveOutput: False - nothing is saved
#             True  - the output directory 'outputDir' will be used to store the
#                     products of this pipeline (intermediate and final).
# When saveOutput is True, nameBasis will be used as basis for the filenames of all outputs
saveOutput = True

# To save to your favourite dir, use : outputDir = "/home/me/myDir/"
outputDir  = str(Configuration.getWorkDir())+"/pacsSpecOut/"
if (not os.path.exists(outputDir)): os.mkdir(outputDir)
if verbose and saveOutput: print "The products of this pipeline will be saved in ",outputDir

# nameBasis will be used as 'basis' for the names of all final fits files
nameBasis  = str(obsid)+"_"+target+"_"+od+"_Hipe_"+hipeVersion+"_calSet_"+calSet+"_"+camera+"_rsrf"
```

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.

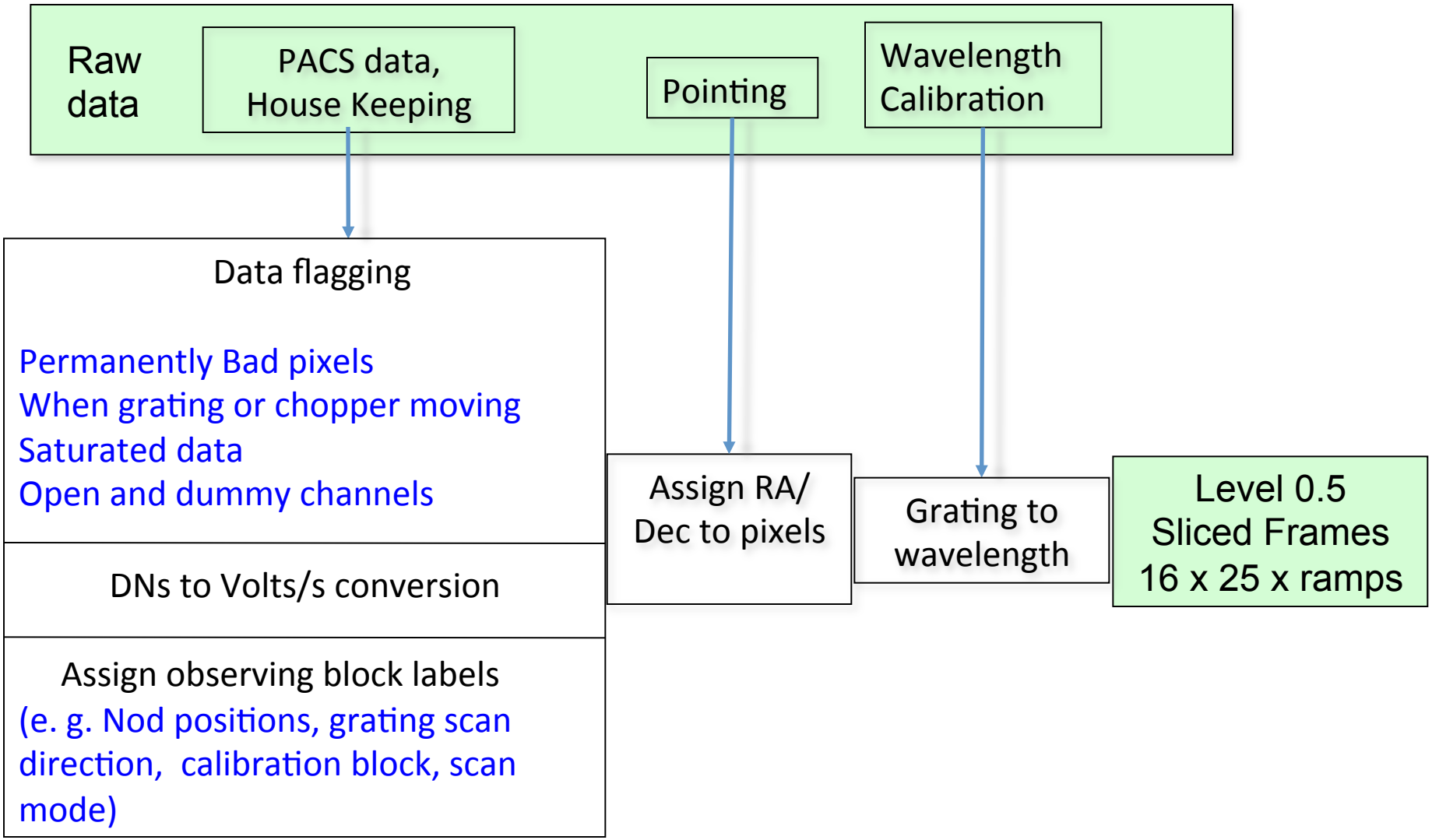


Step 3

Run the 0 → 0.5 pipeline

Basic calibration (pointing, wavelength calibration, slicing)

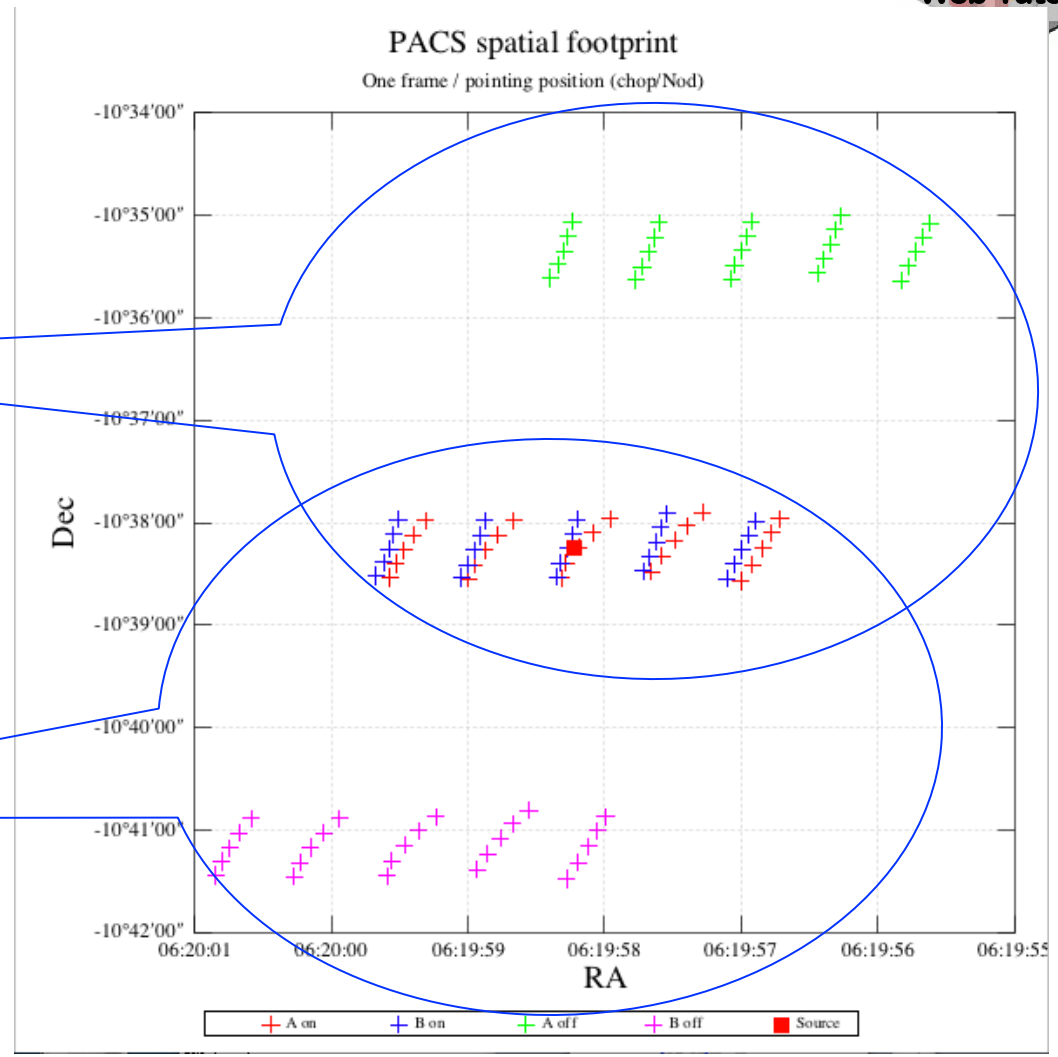
Level 0 → 0.5


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 spatial footprint
if verbose:  ppoint = slicedPlotPointing(slicedFrames, plotBoresight=False)
```

Slicing into nods

The slicing of the data is performed according to rules made explicit in the pipeline. In our example, one line is observed in two nodding positions and with three repetitions. So, we expect 6 slices plus an initial slice containing the calibration block.

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

History | Log | Console

```
noSlices: 7
noCalSlices: 1
noScienceSlices: 6
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"]                     [18,25,1631]
  62.936 - 63.441  both      both
2       true       ["A"]          1         0 0       [2]         ["B3A"]                     [18,25,1631]
  62.936 - 63.441  both      both
3       true       ["A"]          2         0 0       [2]         ["B3A"]                     [18,25,1631]
  62.936 - 63.441  both      both
4       true       ["B"]          2         0 0       [2]         ["B3A"]                     [18,25,1631]
  62.936 - 63.441  both      both
5       true       ["B"]          3         0 0       [2]         ["B3A"]                     [18,25,1631]
  62.936 - 63.441  both      both
6       true       ["A"]          3         0 0       [2]         ["B3A"]                     [18,25,1631]
  62.936 - 63.441  both      both
```

Check: after slicing

7 slices !

Line 1 – B & A nodes – cycle 1

Line 1 – B & A nodes – cycle 2

Line 1 – B & A nodes – cycle 3

```
248
249  # The internal structure of your data has changed
250  if verbose: slicedSummary(slicedFrames)
```

History | Log | Console ×

```
noSlices: 7
noCalSlices: 1
noScienceSlices: 6
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"]
  62.936 - 63.441  both      both
2        true       ["A"]          1         0 0       [2]         ["B3A"]
  62.936 - 63.441  both      both
3        true       ["A"]          2         0 0       [2]         ["B3A"]
  62.936 - 63.441  both      both
4        true       ["B"]          2         0 0       [2]         ["B3A"]
  62.936 - 63.441  both      both
5        true       ["B"]          3         0 0       [2]         ["B3A"]
  62.936 - 63.441  both      both
6        true       ["A"]          3         0 0       [2]         ["B3A"]
  62.936 - 63.441  both      both
```

Continue …

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

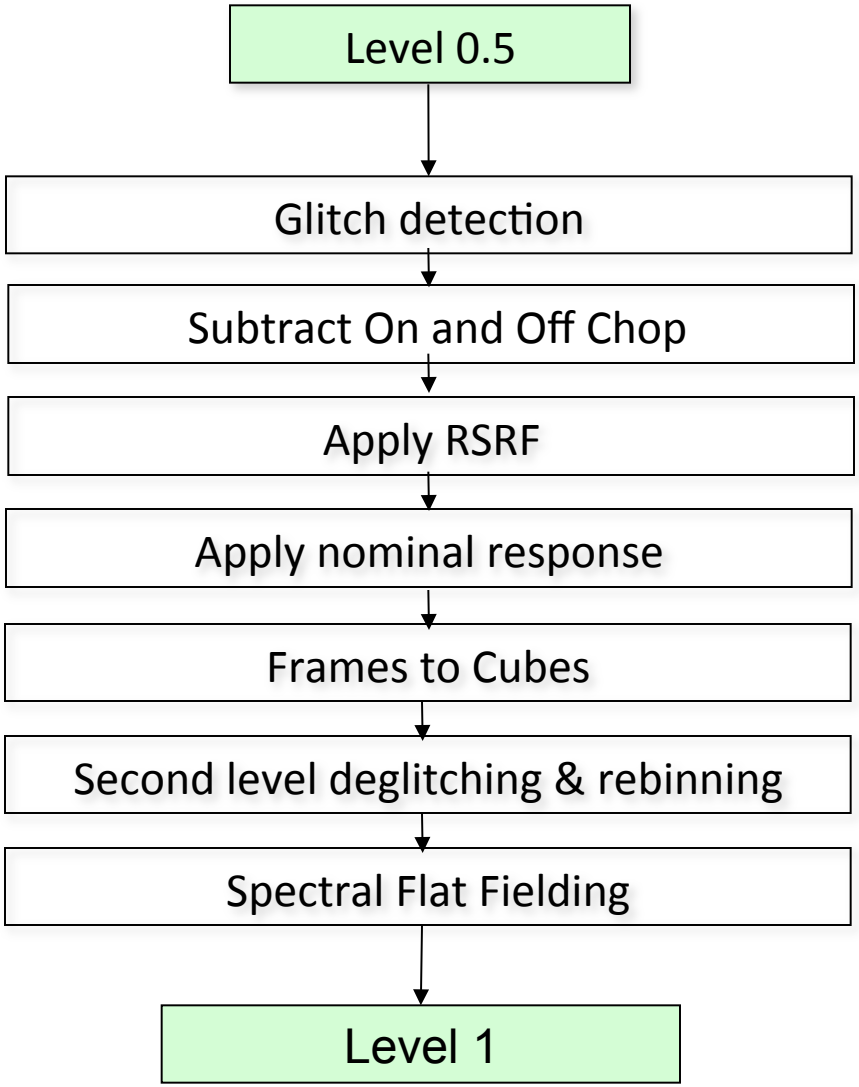
```
# ------------------------------------------------------------------------------
#         Processing      Level 0.5 -> Level 1
# ------------------------------------------------------------------------------
```

Step 4

Run the 0.5 → 1 pipeline

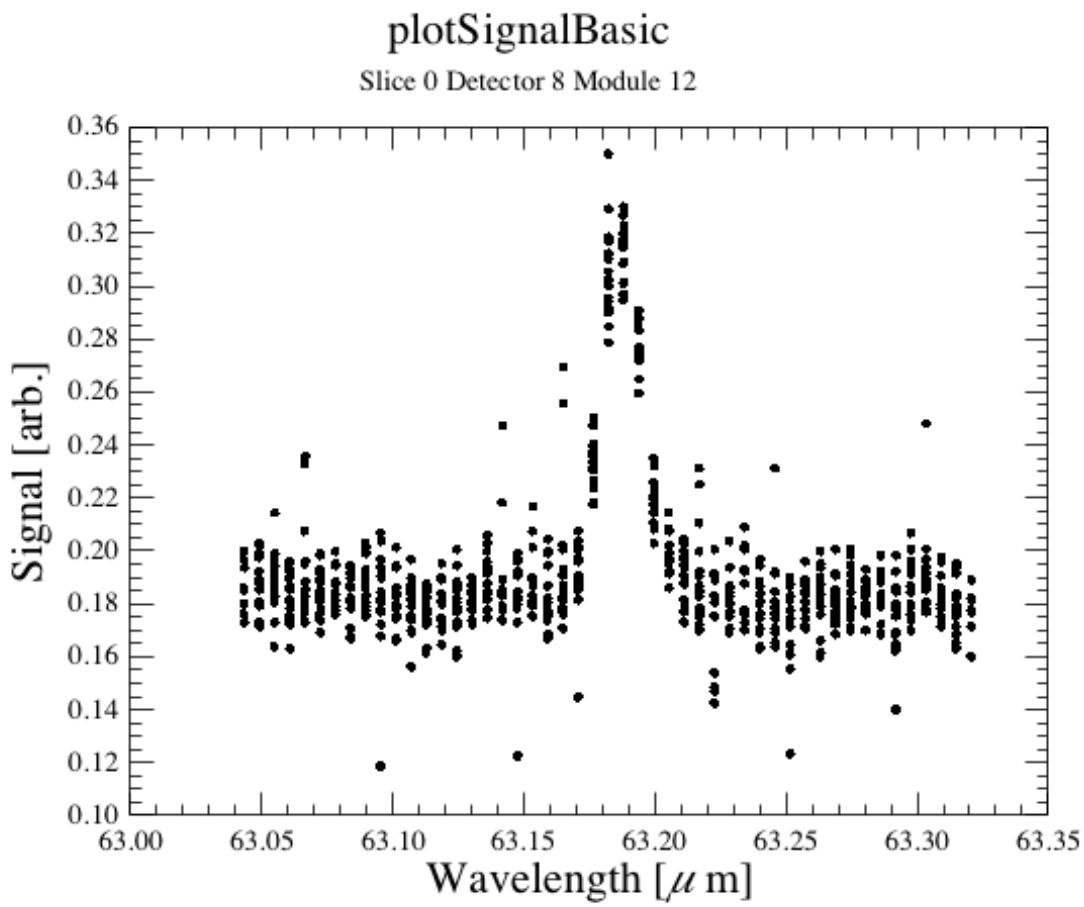
Glitch detection, chop differentiation, RSRF, flat

Level 0.5 → 1

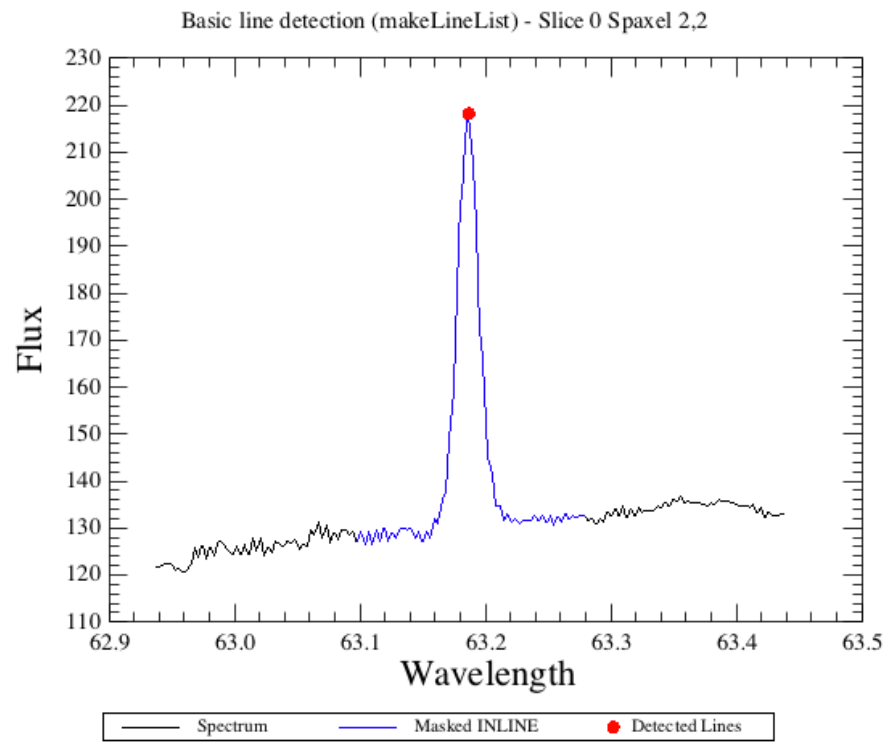

Level 0.5
Glitch detection
Subtract On and Off Chop
Apply RSRF
Apply nominal response
Frames to Cubes
Second level deglitching & rebinning
Spectral Flat Fielding
Level 1

Signal after chop subtraction

Verbose=1 shows
The data are only on the ON position (OFF being subtracted)

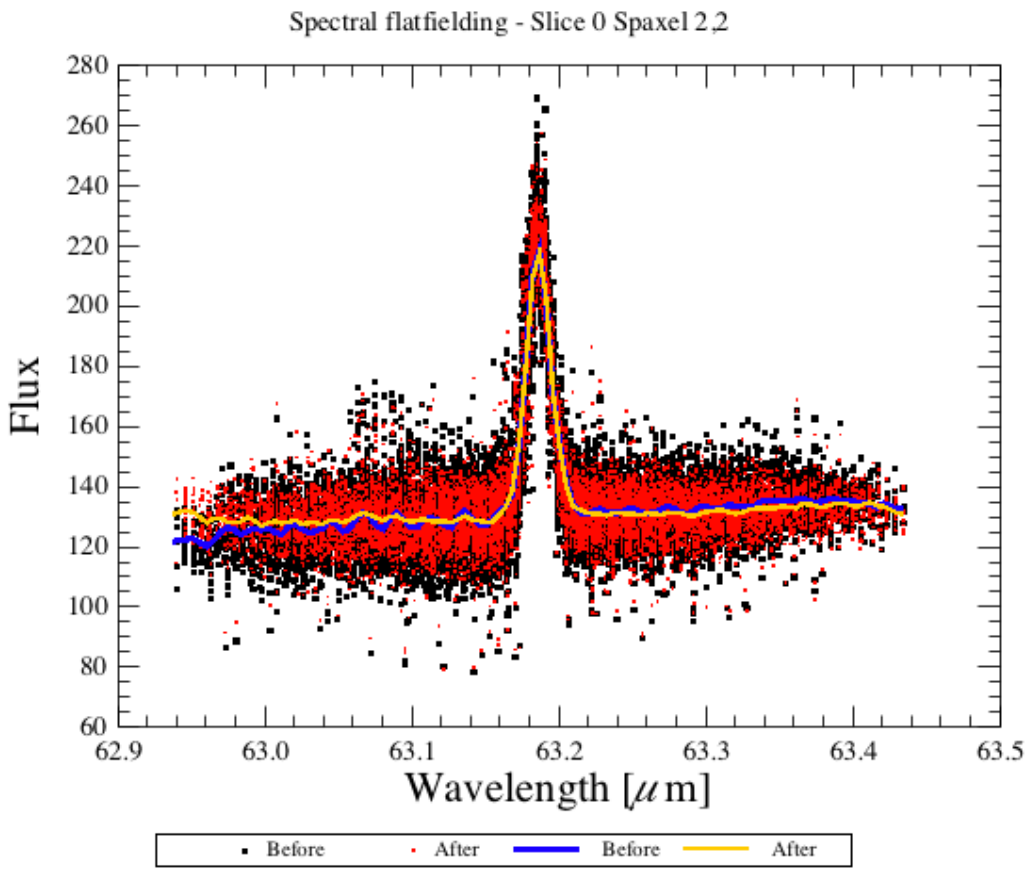


Check: Spectral FlatField



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 = [63.187], 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
# ------------------------------------------------------------------------------
```