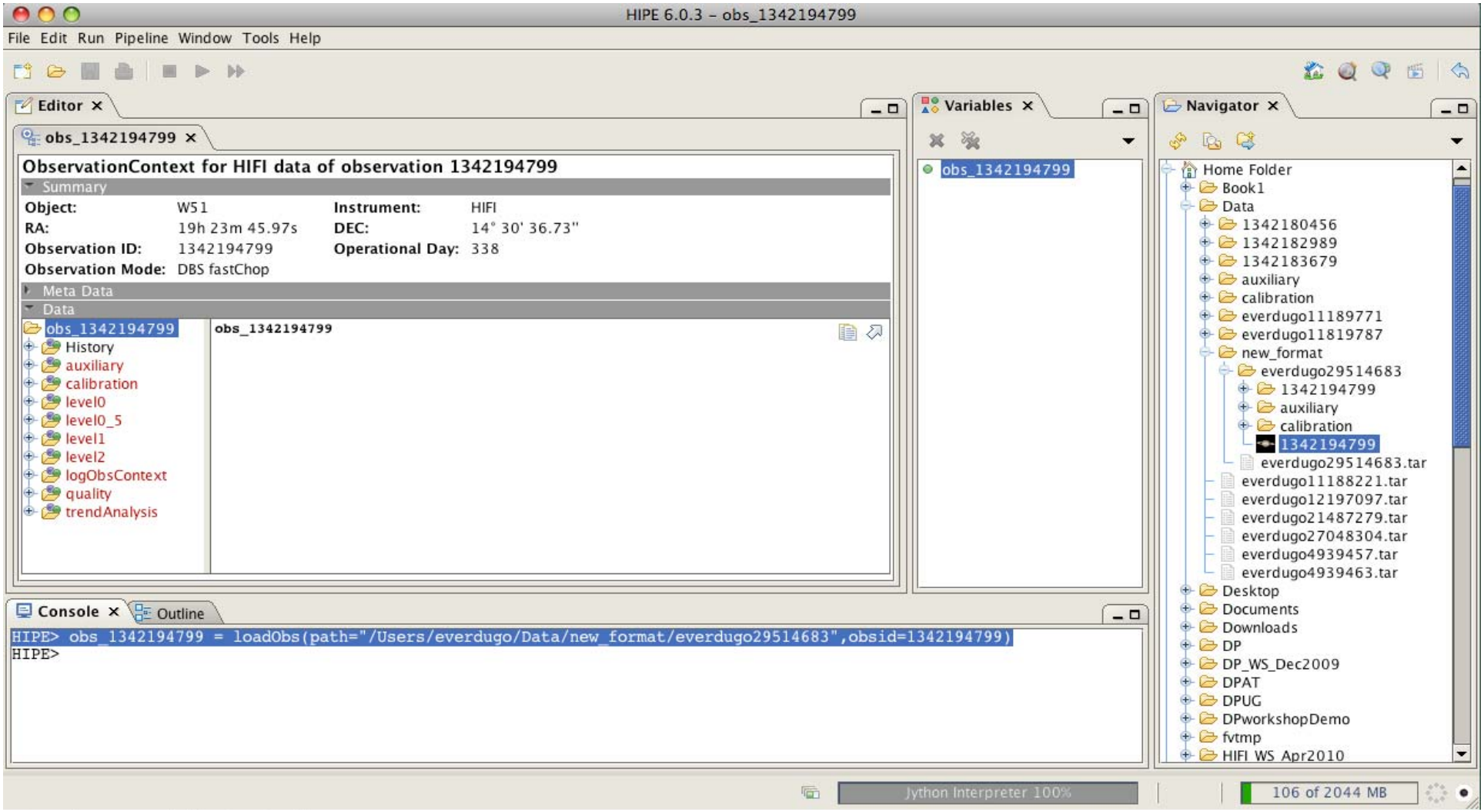
Data Access & Data Handling in HIPE

Eva Verdugo
HSC-ESAC

HERSCHEL SPACE OBSERVATORY

Read Retrieved data from HSA in HIPE

- From the Navigator view: double click on the obs. xml file
- A variable is created and showed in the editor
- The command used is echoed in the console: ***loadObs***



Read Retrieved data from HSA in HIPE

- Interactively: From a GUI
  - Double click on the observation in the Navigator view
- Command line:
  - Task: ***loadObs***
    - ✓ path to the data
    - ✓ obsid (only required if more than 1 obsid under the same path)

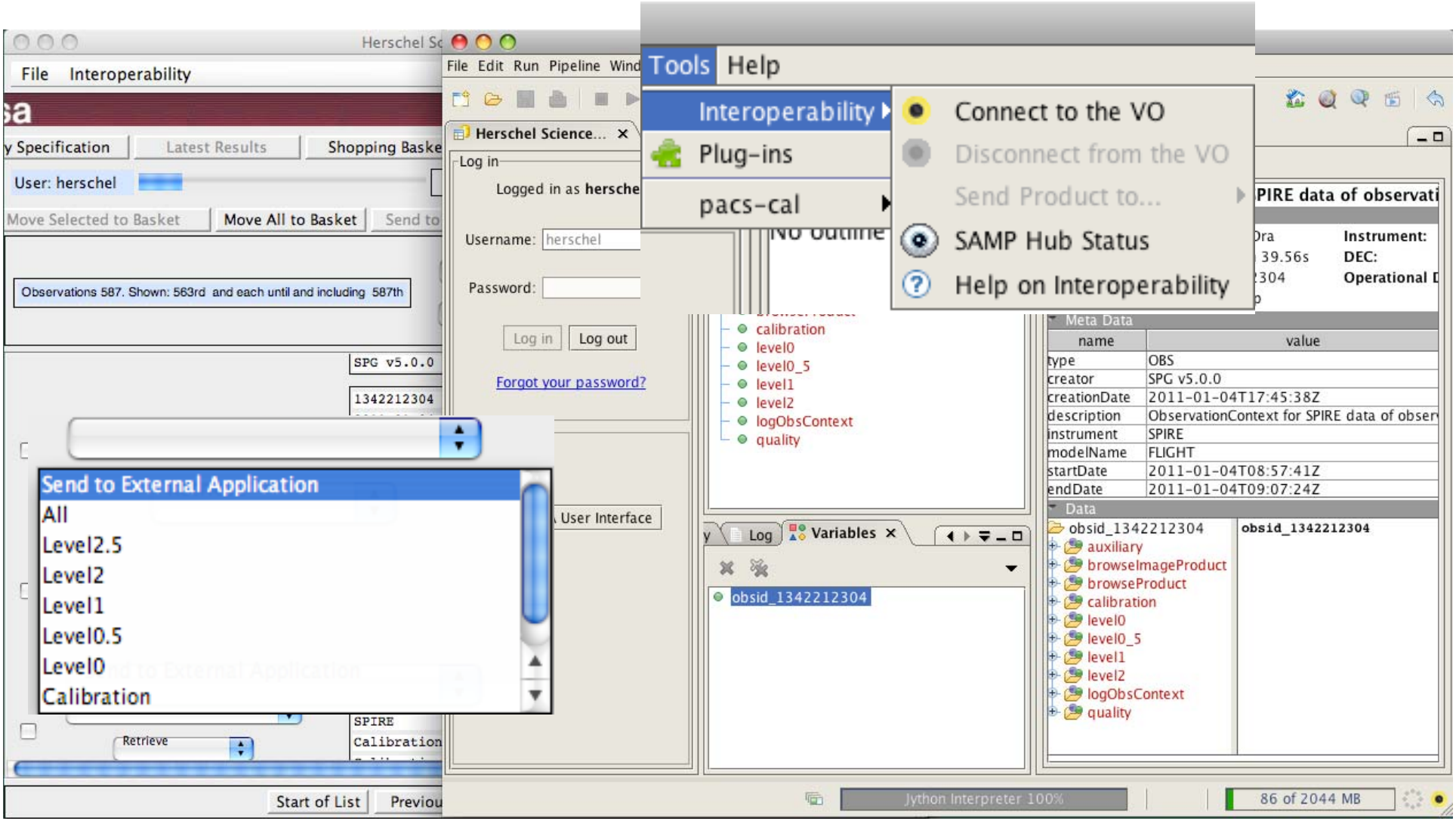
obs = ***loadObs***(“/Users/everdugo/Data/everdugo29514683”,[obsid])

It can be a whole obsContext or a partial observation (supported since HIPE 6.0)

HERSCHEL SPACE OBSERVATORY

A quick look of the Data

Interactively: "***Send to External Application***" from HUI



A quick look of the Data

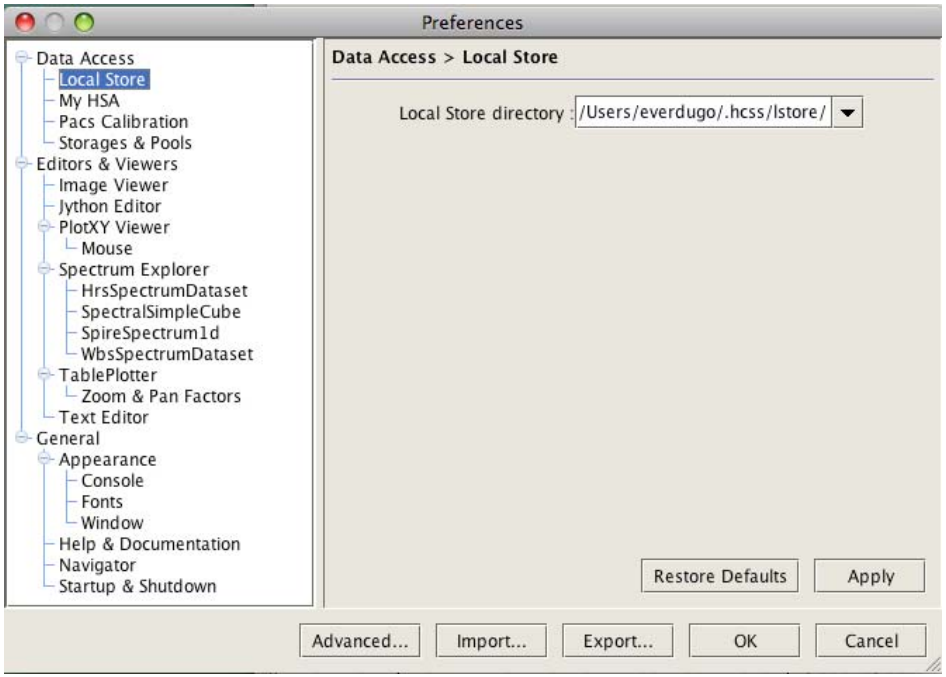
- Interactively: Send to External Application
  - Open HUI & HIPE (or HUI from HIPE)
  - HIPE => Tools => Connect to the VO
  - HIPE => Herschel Science Archive Perspective => Login
  - HUI => Query Results => ***Send to External Application***
  - A variable obsid_xxxxxx is created in HIPE and displayed in the Editor
  - The data is **NOT** in your disk; **Only in HIPE memory**

- Command line:
  - Task: ***getObservation***
    - ✓ obsid
    - ✓ useHSA = True

      obs = ***getObservation***(134222305,useHsa=True)
  - A variable obs is created in HIPE
  - The data is **NOT** in your disk; **Only in HIPE memory**

Advance usage: Pools

- Pools are “special” directories in your disk (local stores)
- By default they are created under .hcss/lstore/



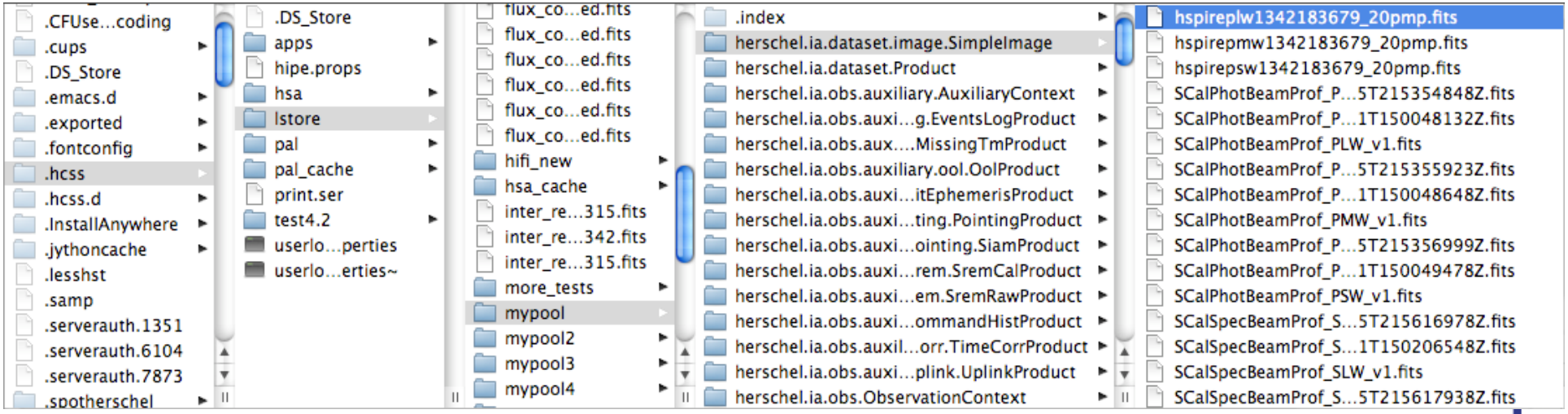
Use Edit ➔ Preferences
to change default location

- They constitute a <u>local database</u> which implies pools should not be
  - Moved
  - Renamed
  - Partially deleted
  - Manipulated manually

HERSCHEL SPACE OBSERVATORY

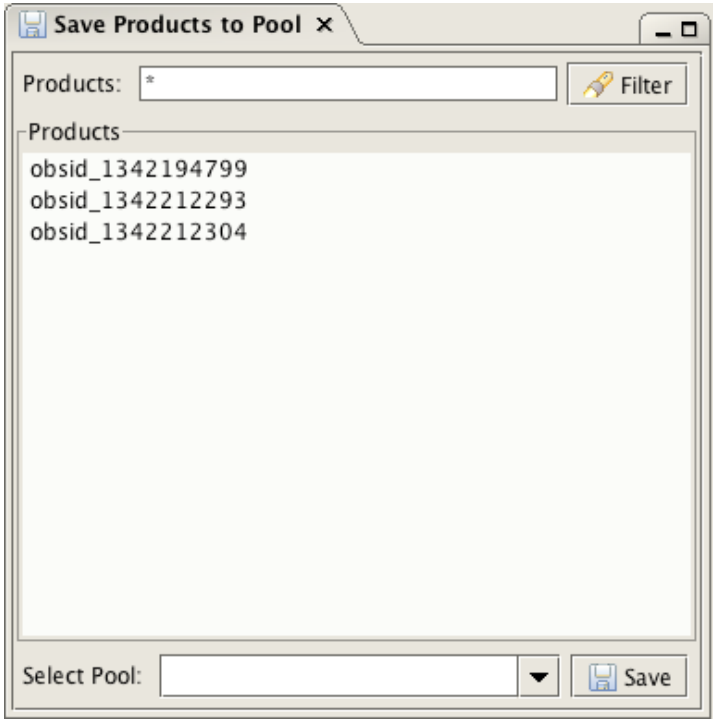
Data Structure in Pools

- Ordered by “classes” under which the FITS files are stored
- HIPE does the tree-like structure from this

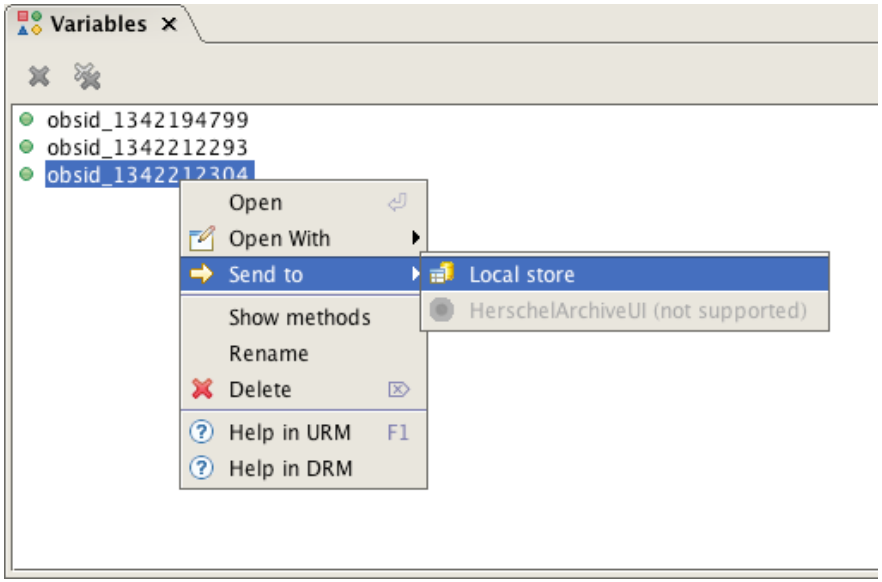


How to populate Pools

- Interactively: Save Products to Pool View
  - Displays all product-type variables of your HIPE session
  - Save into an existing pool or create a new one



- Right click on the variables



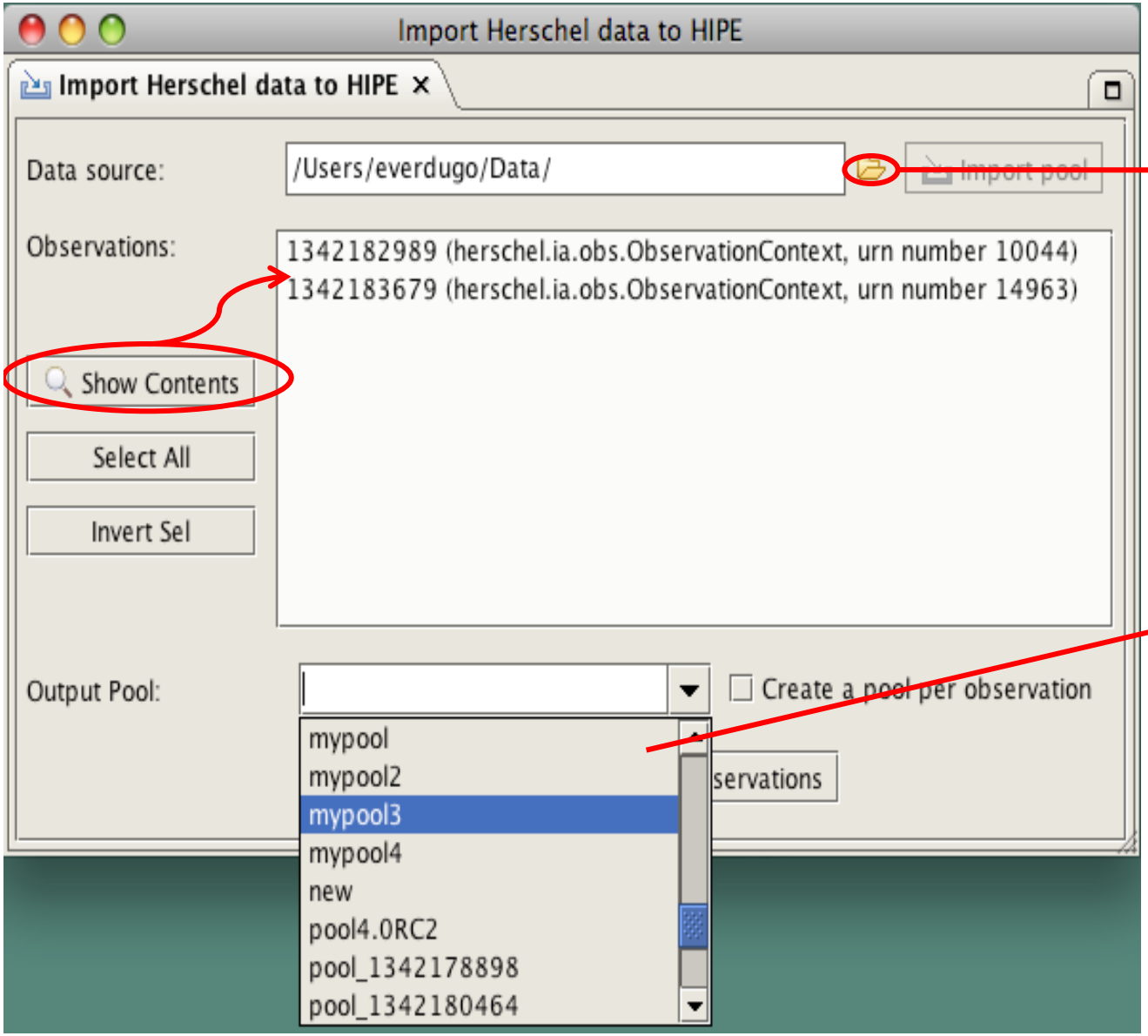
- Command Line: ***saveObservation***

  saveObservation(obs)

  saveObservation(obs, poolName="od144")

Saving data retrieved from HSA into pools

Import Herschel data to HIPE view



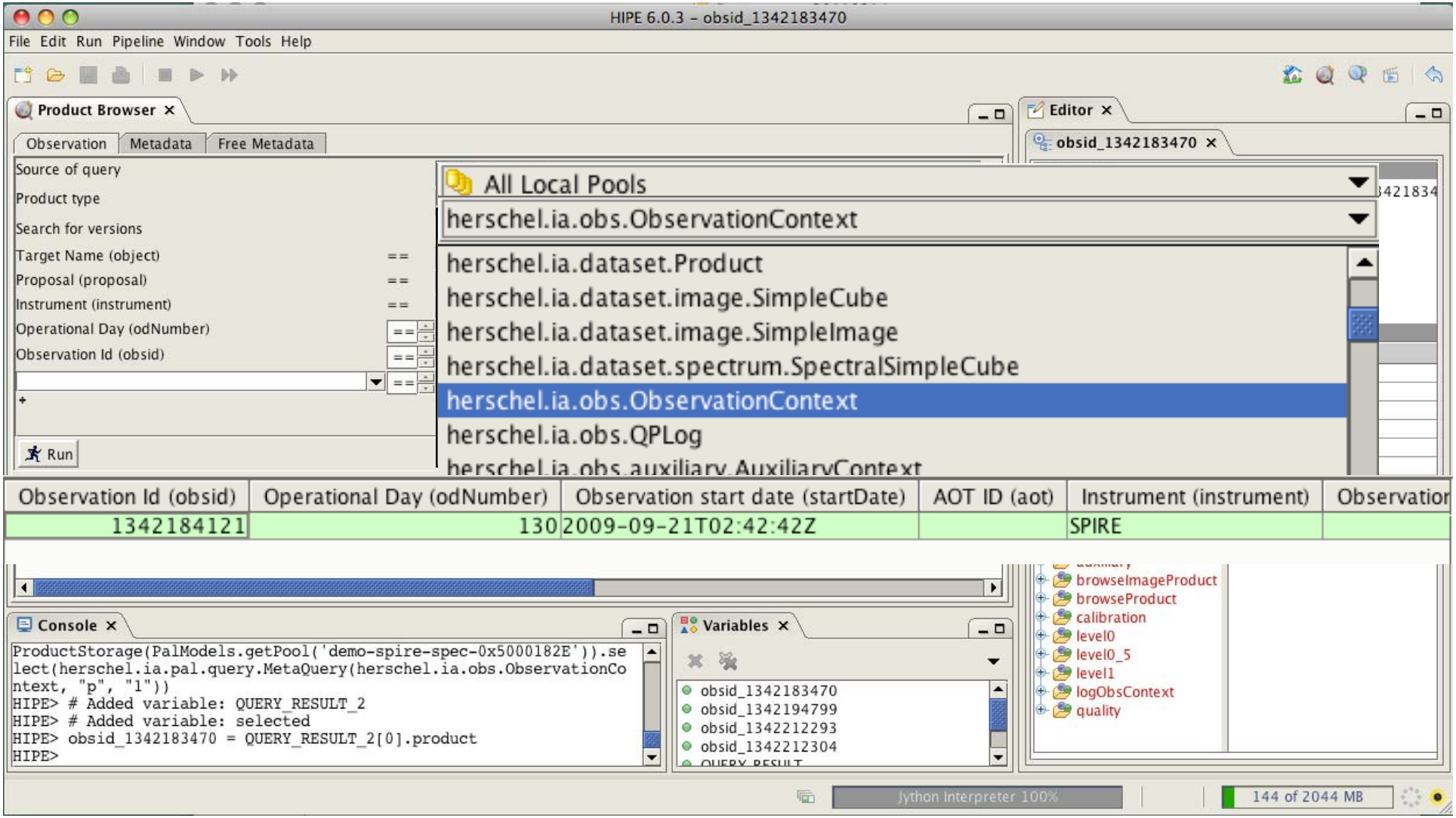
Select a directory with HSA data

Select an output pool or create a new one

HERSCHEL SPACE OBSERVATORY

Access Data in pools: Interactively

Product Browser Perspective



Access Data in pools: Command line

- Access pools in scripts is done with the task:

getObservation(obsid, poolName)

- In this workshop we have provided you with pools that will be read by the pipeline scripts with ***getObservation***

HERSCHEL SPACE OBSERVATORY

Powerful queries using pools

- Interactively: with the Product Browser
  - On pre-defined metadata
  - On free metadata

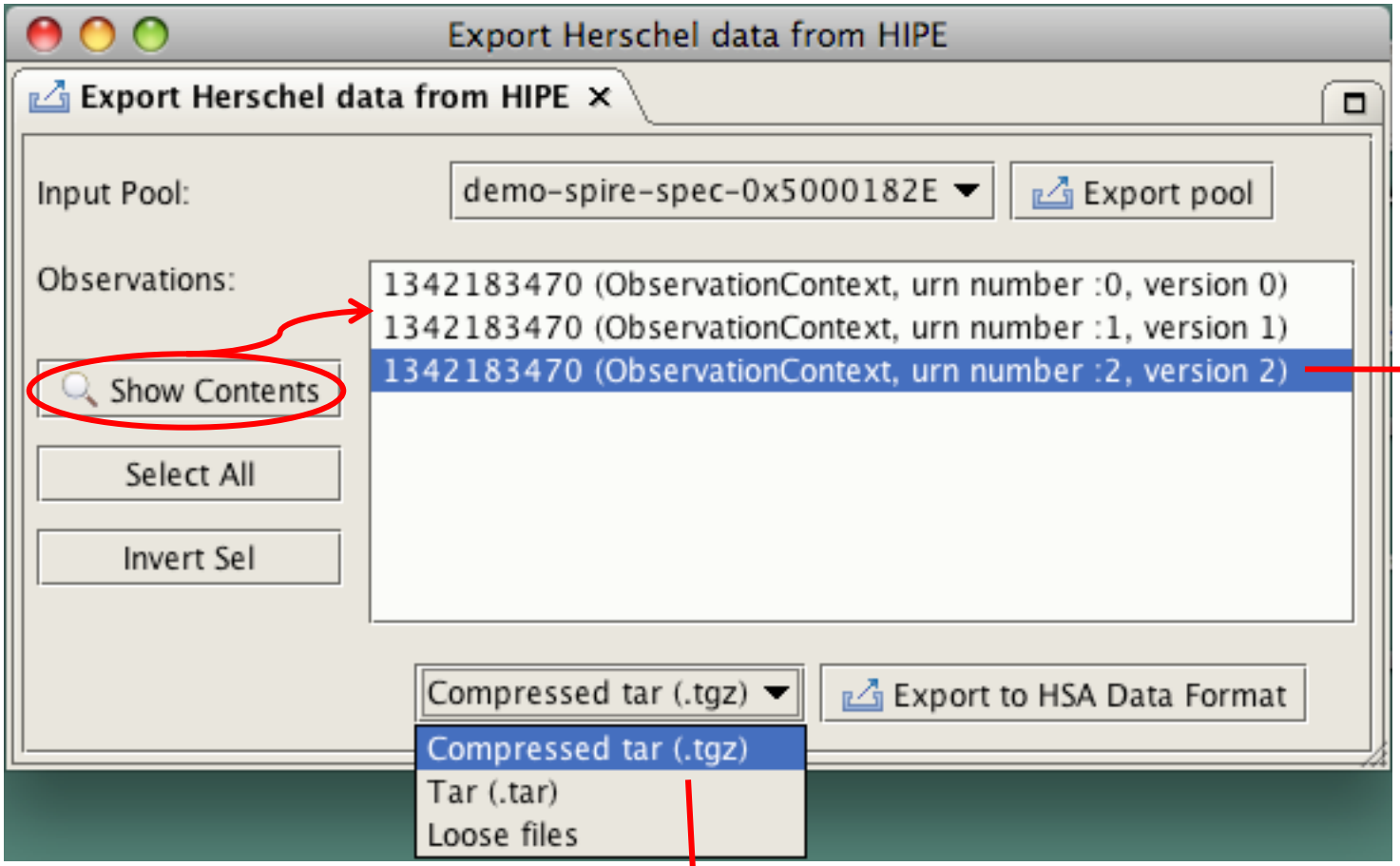
- Scripts: **Possible queries down to data level in pools**

  The following demo script searches for all simpleImage products in a pool and retrieve only those images in which at least one pixel has a flux value larger than 1 Jy. It does this in a couple of seconds for 32 simpleImages.

```
# Demo script to make PAL searches down to data level
# Author: Paul Balm (HSC)

# preparatory work:
# prepare a pool called dpug_search with all scanmaps in OD 160
# search to data level in dpug_search pool

# Define a function that returns true for the images we're looking for
def testImage(i):
        # image type: Blue detector
        if i.type != "HPPPMAPBS":
                return False
        # AND: at least one pixel over 1 Jy
        if max(i.image) < 1.0:
                return False
        return True

 # Define the query using this function and run it
query = FullQuery(SimpleImage,'i','testImage(i)')
refs = dpug_store.select(query)

 # Print the obsids of the found images (it should retrieve the 6 obs of the asteroid 4 Vesta)
 # which are the only ones with fluxes above 1 jy.
 i = 0
for ref in refs:
        m = ref.meta
        print str(i),': Obsid',m['obsid'].value
        i += 1
```

Export Herschel Data in pools

• Data saved in pools can be also exported to the tree-like structure equal to the one provided by the HSA:



Choose version

Choose output

Summary: Lifecycle of an observation

1. Retrieve the observation(s) from the HUI: Whole or partial

2. Read it into HIPE: obs= ***loadObs("path")***

3. Process the observation with your favourite pipeline script

4. Save:
   a) Individual FITS files: ***simpleFitsWriter***
   b) Replace new files in the same obsContext and save into a pool:
      obs.level2=xxxx
      saveObservation(obs,"pool")
   c) Save new version of the obsContext into a pool
      saveObservation(newobs,"pool")

5. Export obs to tree-like structure: ***Export Herschel Data from HIPE***

HERSCHEL SPACE OBSERVATORY