Mining the HSA using scripts

David Shupe, NHSC

...based on Roland Vavrek's presentation at HIPE Forum 2012

These are the kinds of use cases we want to handle

- “Find all spectral cubes for PACS Line Spectroscopy Mapping that cover the [OI] 63um line, and display the five with highest SNR.”
- “For 13 famous AGB stars, find whether there are HIFI observations covering excited CO (Ju > 8).”
- “For my list of targets, find which have PACS observations at 70 microns.”
- “Make a list of nearby galaxies observed in the [CII] 158 um line.”

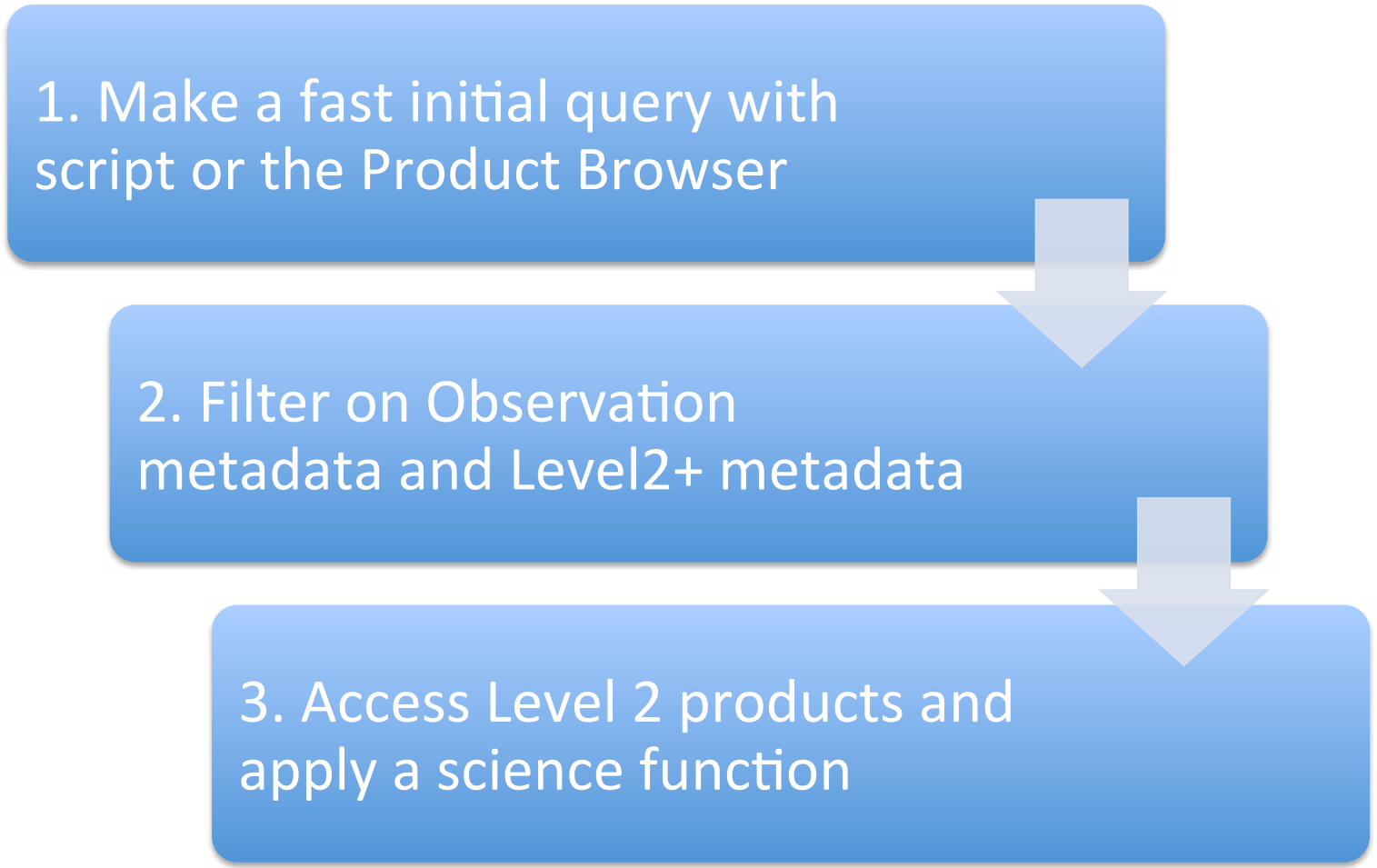
Techniques used along the way

- Make a fast initial query of the HSA
- Manipulate the resulting ProductReferences
- Write filter functions for metadata
- Search on metadata disallowed by the HSA in observations and Level-2 products
- Apply a science function to the data to pull out e.g. line SNR
- Build up a TableDataset
- Store the results of these searches

Limitations on queries of the HSA prevent searching everything in one pass

- Only a limited set of metadata are allowed to be queried
  - See http://hsa.esac.esa.int:8080/aio/doc/alias.html
  - Most are not useful in actual data
- Hard limit on number of results
  - 5,000 or 25,000

Solution: traverse the results of a query just as you'd inspect an observation

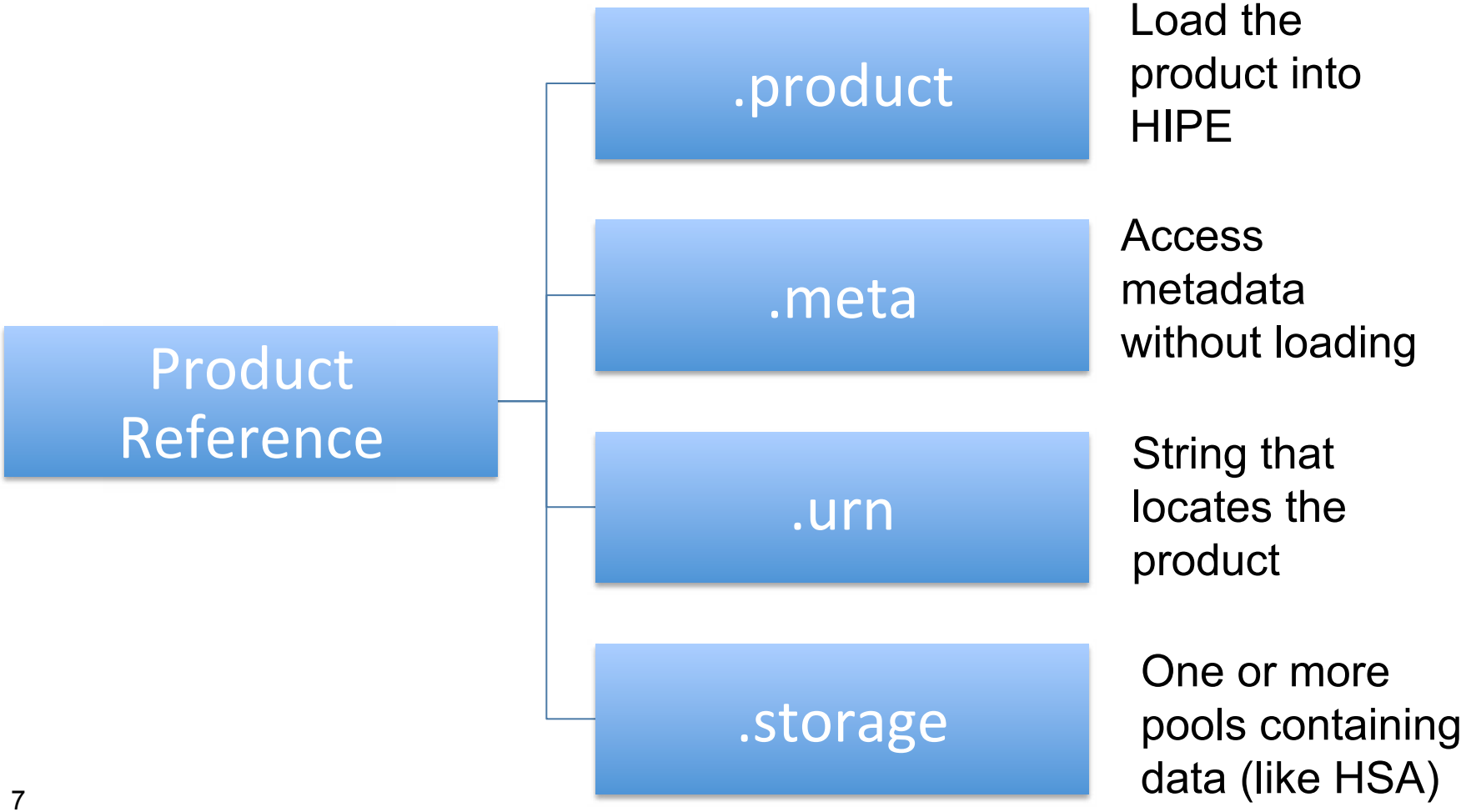


Step 1: Make a fast query of the metadata that are allowed

- Search metadata of all observations:
  instrument==‘PACS’ and
  aot==‘Line Spectroscopy’ and
  obsMode==‘Mapping’

- Returns 914 results in 30 seconds

The query returns a StorageResult containing ProductReferences



Step 2a: Define a filter function for disallowed metadata in ObsContext

- Check that metadata is accessible
  - If not, the observation is not public
- Test the Observation Context metadata

```
obsref.meta['lineStep'].value <= 4.5
    and
obsref.meta['pointStep'].value <= 4.5):
```

- Return obsref if valid, None otherwise

Step 2b: Define a filter function for metadata in Level 2 product

- def pacsSpecLevel2Filter(obsref, type='HPS3DPB', minWave=62.0, maxWave=65.0):

- Test that wavelengths fall in these limits (for [OI] 63 micron line)

Step 2c: Loop over all the references in fast query result, and apply filters

- def filterObsAndLevel2(storageresult, obsfilter,level2filter):

- Takes about 35 minutes

After the query is done, convert the list of reference to URN strings

```
obsurns = [ref.urn for ref in obsreflist]
l2OIurns = [ref.urn for ref in level2reflist]
```

- ProductReferences and StorageResults cannot be saved and restored fully

- URNs are just strings so they store well

Step 3a: Make a list of references to the cubes, and save metadata in a table

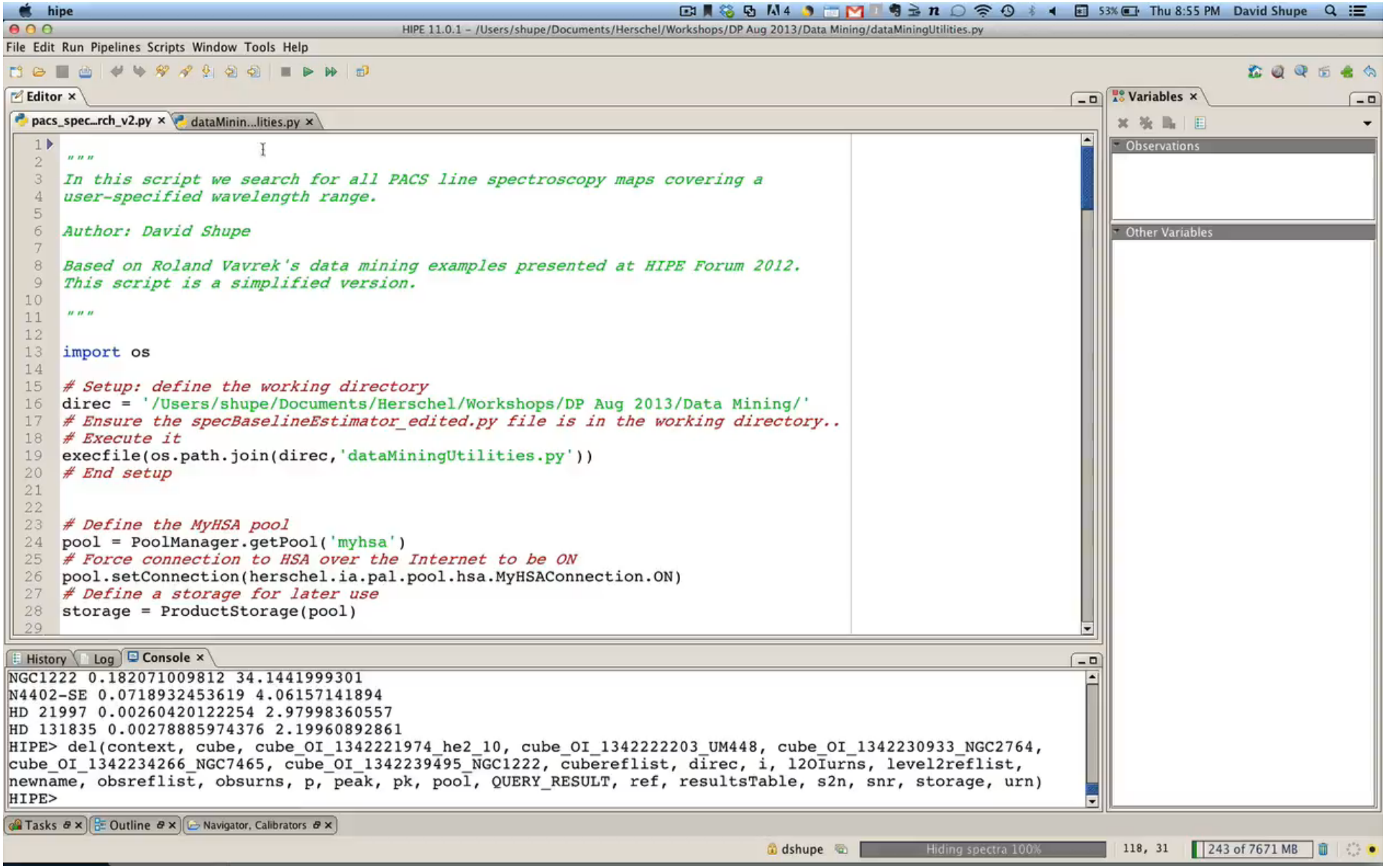
```
cubereflist = []
for ref in level2reflist:
    context = ref.product
    for p in context.refs:
        cubereflist.append(p)

resultsTable = tableFromMetadata(cubereflist,\
  ['obsid','object','proposal','ra','dec','redshiftValue',\
      'redshiftType','lines'])
```

Step 3b: Access the cube data and apply a “science function”

- Science function = quickCubeSnr
  - Fast, rough estimate from extracted spectrum
    - Noise= standard deviation of flux
    - Peak=85% level of flux

- Add the SNR and peak to the results table

Finishing: Display the 5 cubes with highest SNR, and save the results table


hipe
Thu 8:55 PM David Shupe
HIPE 11.0.1 - /Users/shupe/Documents/Herschel/Workshops/DP Aug 2013/Data Mining/dataMiningUtilities.py
File Edit Run Pipelines Scripts Window Tools Help
Editor
pacs_spec...rch_v2.py
dataMinin...lities.py
"""
In this script we search for all PACS line spectroscopy maps covering a
user-specified wavelength range.
Author: David Shupe
Based on Roland Vavrek's data mining examples presented at HIPE Forum 2012.
This script is a simplified version.
"""
import os
# Setup: define the working directory
direc = '/Users/shupe/Documents/Herschel/Workshops/DP Aug 2013/Data Mining/'
# Ensure the specBaselineEstimator_edited.py file is in the working directory..
# Execute it
execfile(os.path.join(direc,'dataMiningUtilities.py'))
# End setup
# Define the MyHSA pool
pool = PoolManager.getPool('myhsa')
# Force connection to HSA over the Internet to be ON
pool.setConnection(herschel.ia.pal.pool.hsa.MyHSAConnection.ON)
# Define a storage for later use
storage = ProductStorage(pool)
Variables
Observations
Other Variables
History
Log
Console
NGC1222 0.182071009812 34.1441999301
N4402-SE 0.0718932453619 4.06157141894
HD 21997 0.00260420122254 2.97998360557
HD 131835 0.00278885974376 2.19960892861
HIPE> del(context, cube, cube_OI_1342221974_he2_10, cube_OI_1342222203_UM448, cube_OI_1342230933_NGC2764,
cube_OI_1342234266_NGC7465, cube_OI_1342239495_NGC1222, cubereflist, direc, i, l2OIurns, level2reflist,
newname, obsreflist, obsurns, p, peak, pk, pool, QUERY_RESULT, ref, resultsTable, s2n, snr, storage, urn)
HIPE>
Tasks
Outline
Navigator, Calibrators
dshupe
Hiding spectra 100%
118, 31
243 of 7671 MB

If you have a list of targets, you can search the observation log for OBSIDs

- https://nhscsci.ipac.caltech.edu/sc/index.php/ObsStatus/ObservingLogTables
  - Observing log files (compiled by B. Schulz)
  - Scripts for searching by position or name
  - Warning: calibration observations aren't included
- The HSC will export the log to Vizier

- See ConeSearchObsTable.py script for an example involving famous AGB stars
  - Get OBSIDs and then list of references to observations
  - Proceed as for PACS Line Spectroscopy example

…happy hunting!