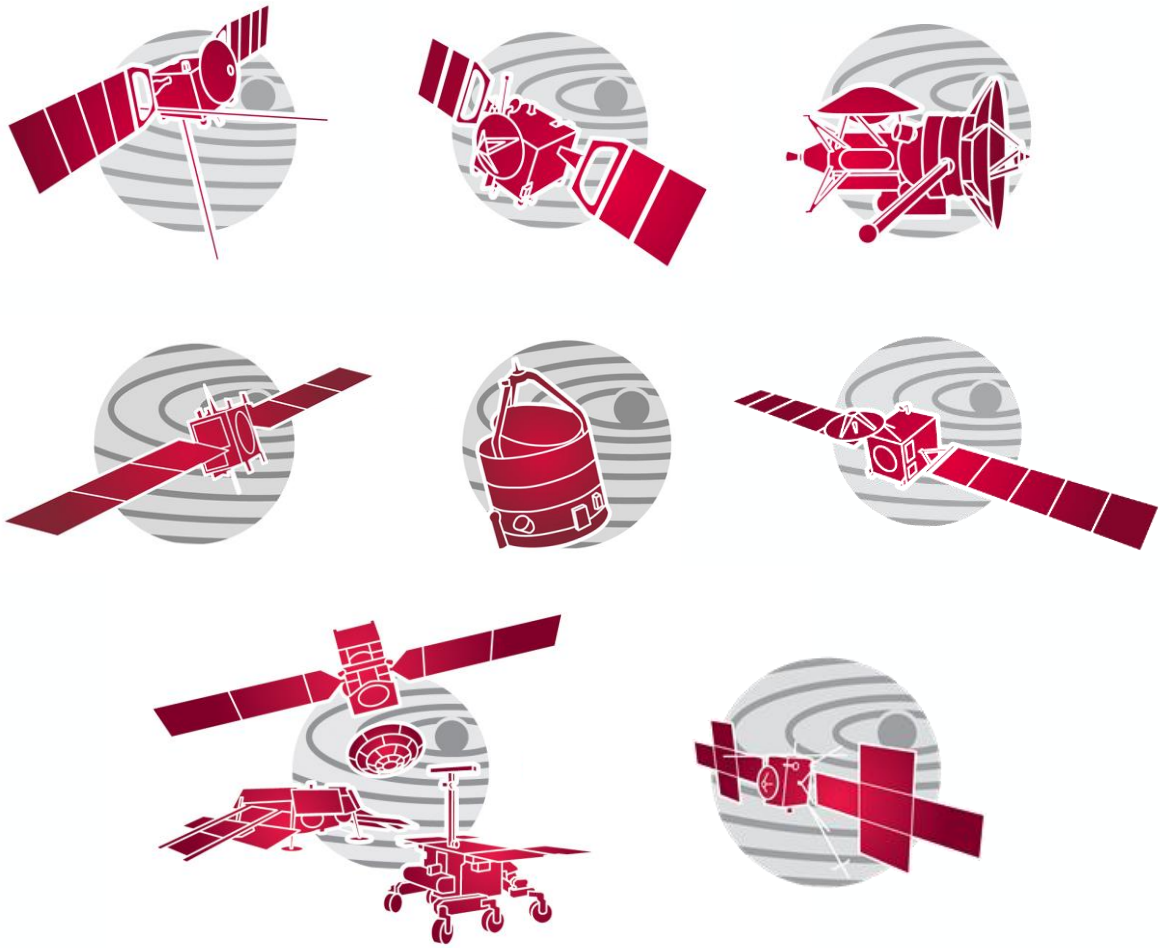
DataLabs for the Planetary Science Archive

Mark S. Bentley
on behalf of the PSA team

06/10/2022

Reminder: What is the PSA

- The Planetary Science Archive (PSA) is:
  - a multi-mission science archive
    - for ESA's planetary missions
    - orbiters, landers, rovers, etc.
- Long term archive
  - data should be usable in 50 years
- But also used operationally
  - PDS4 missions deliver ~daily
  - data private until reviewed and released
  - (data volumes have a mix of public/private data)
- Adopts the NASA PDS standard
  - missions delivering PDS3 and 4
  - these "wrap" many standard formats
  - extensive meta-data based on an information model



Use cases for DataLabs

The scope for using DataLabs is clearly enormous, and we won't know what the community will do with it until we open it up and provide the relevant tools and APIs. However, the first items likely to be of use are:

- Data Tutorials
  - PSA contains data from many different instrument types, stored in many different ways
  - User feedback is that **finding** data is easy, but working with it is hard
  - One way to help is to provide tutorials in Jupyter Labs or similar, e.g.
    - replicate the analysis from a paper
    - repeat the calibration from raw to calibrated, or derived
  - DataLabs works well because we can provide a working environment
    - and users can copy/paste and develop their own more complex workflow from here
- Work with arbitrary meta-data
  - PSA only indexes a small fractions of the tens of thousands of meta-data attributes
  - DataLabs allows databases to be built/used without download thousands of data product
    - note that this may only be a temporary use case until we have all meta-data available

What do we need from DataLabs?

- In the first instance, the main low hanging fruit needs only Jupyter Labs
  - plus an appropriate python environment
  - fortunately the superset of useful packages is fairly constrained

- Note two extremely useful open source packages for reading actual data:
  - PDS3: https://github.com/MillionConcepts/pdr
  - PDS4: https://github.com/Small-Bodies-Node/pds4_tools

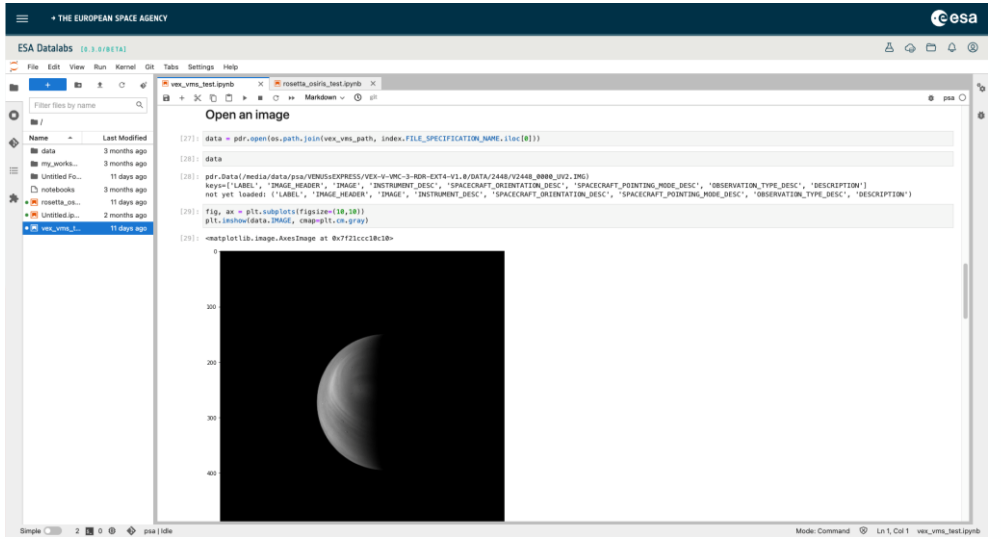
- The examples shown today use the generic ESDC Jupyter Labs DataLab
  - with a “psa” conda environment loaded with the relevant packages

- Moving forward we need to engage the community to see what else they may need

Approaches to DataLabs integration

The PSA is following this approach for integrating data with DataLabs:

- Internally moved data from legacy missions to two volumes (legacy + Rosetta) and mount them
    - ongoing missions with a mix of public/private data will be handled separately
        - discussion needed – internally we are setting up FUSE for this purpose
- A new TAP column will be added with the path to each data product
    - needs a stable mount point in a PSA DataLab
    - allows users to query and load data dynamically
- We need a PSA DataLab
    - Jupyter Labs + common python tools
        - pdr, pds4_tools etc
    - currently using generic lab for testing
- Eventually may link UI to DataLabs
    - e.g. “send to DataLabs” option

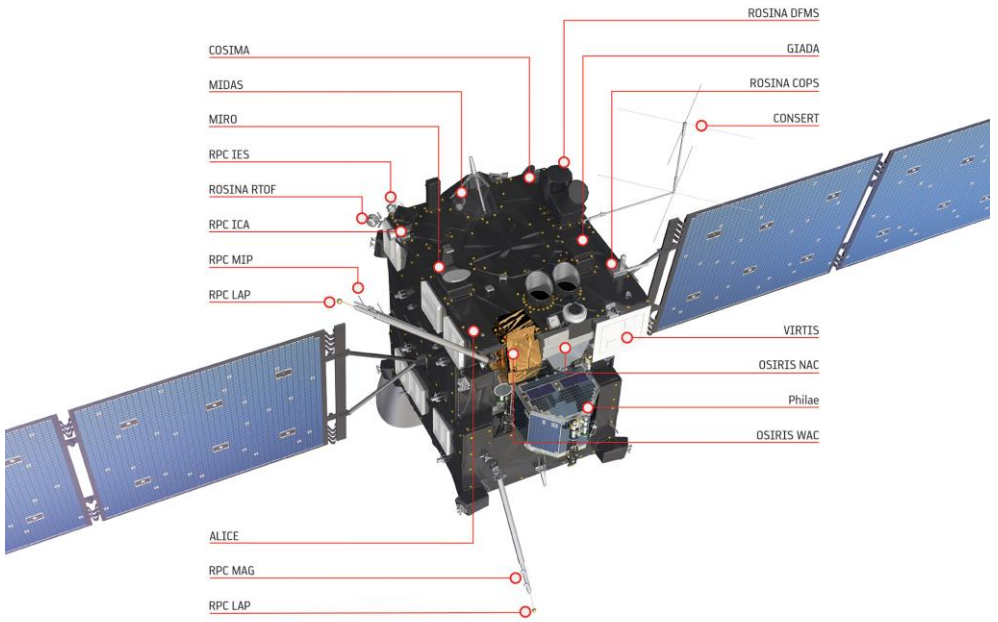


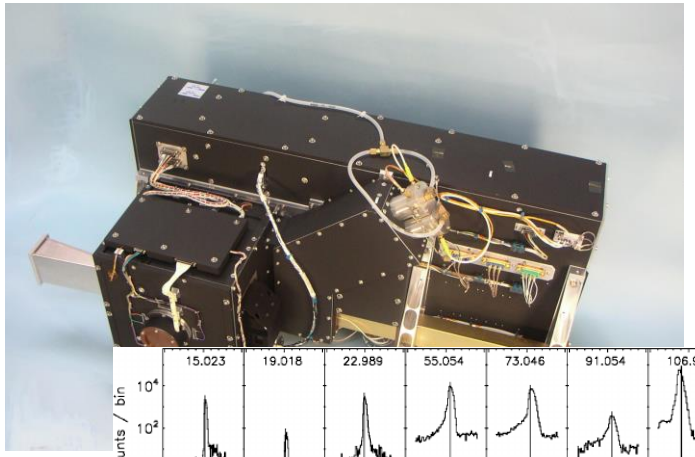
Coming soon – PDS registry integration

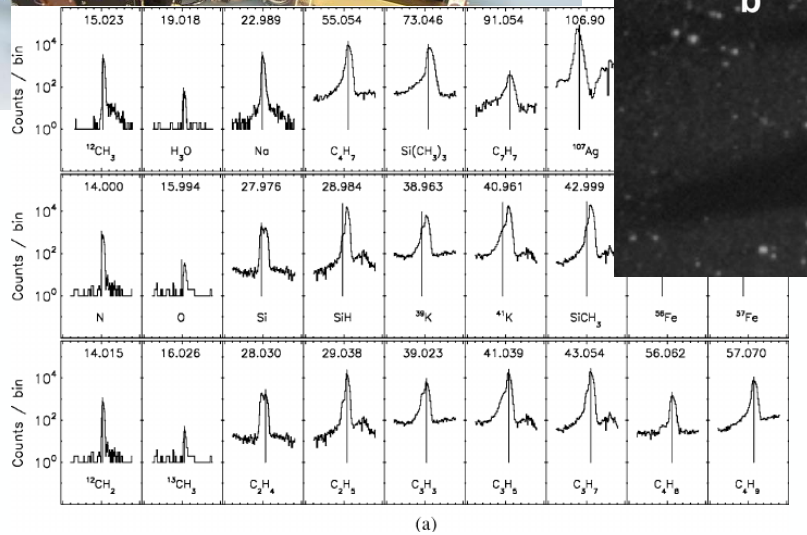
- PSA plans to ingest all PDS4 meta-data into the PDS registry
  - an AWS-hosted OpenSearch instance
  - structured using the PDS information model
- This will enable use of the PDS API on PSA data
  - including full meta-data search/access which is a highly requested feature
- DataLabs/Jupyter notebooks may be updated to use the API
  - as an alternative to TAP, allowing queries of arbitrary meta-data
- TBD if we need a local copy of the registry in-house, sync'd with the PDS one
  - for local searches from PSA UI and/or DataLabs
  - and to map the datasource to a local path rather than https URL
    - or we do that translation on the fly

Examples: Rosetta/COSIMA

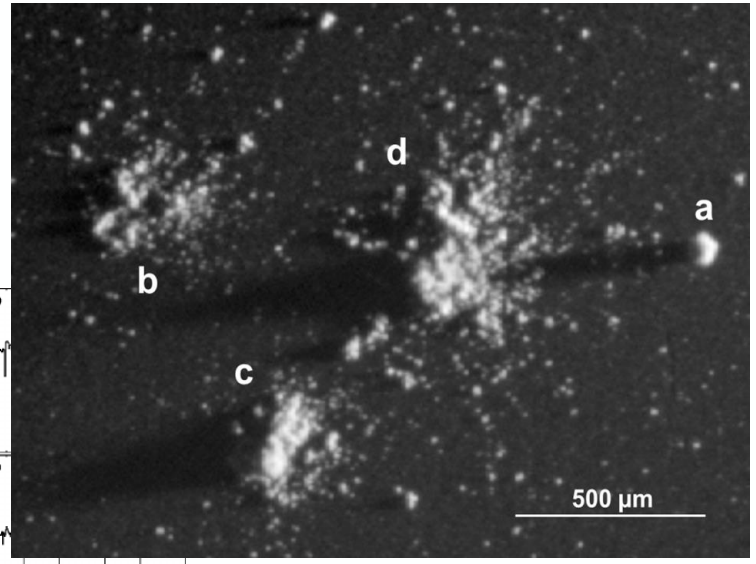
- This example was prepared by Maddalena Buggati as part of her internship
  - and updated slightly for this workshop
  - it uses data from the COSIMA microscope and dust mass spectrometer on the Rosetta orbiter



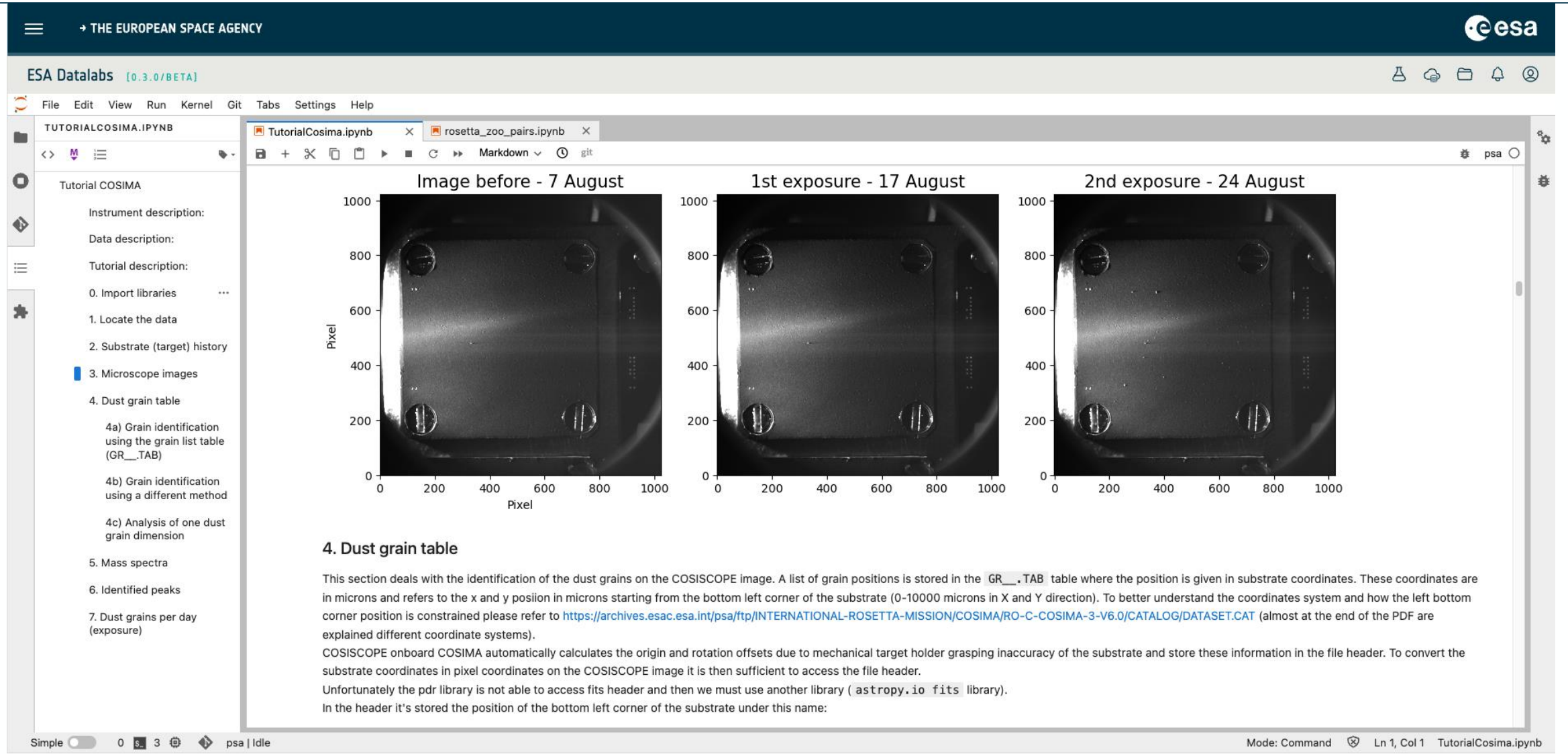




(a)



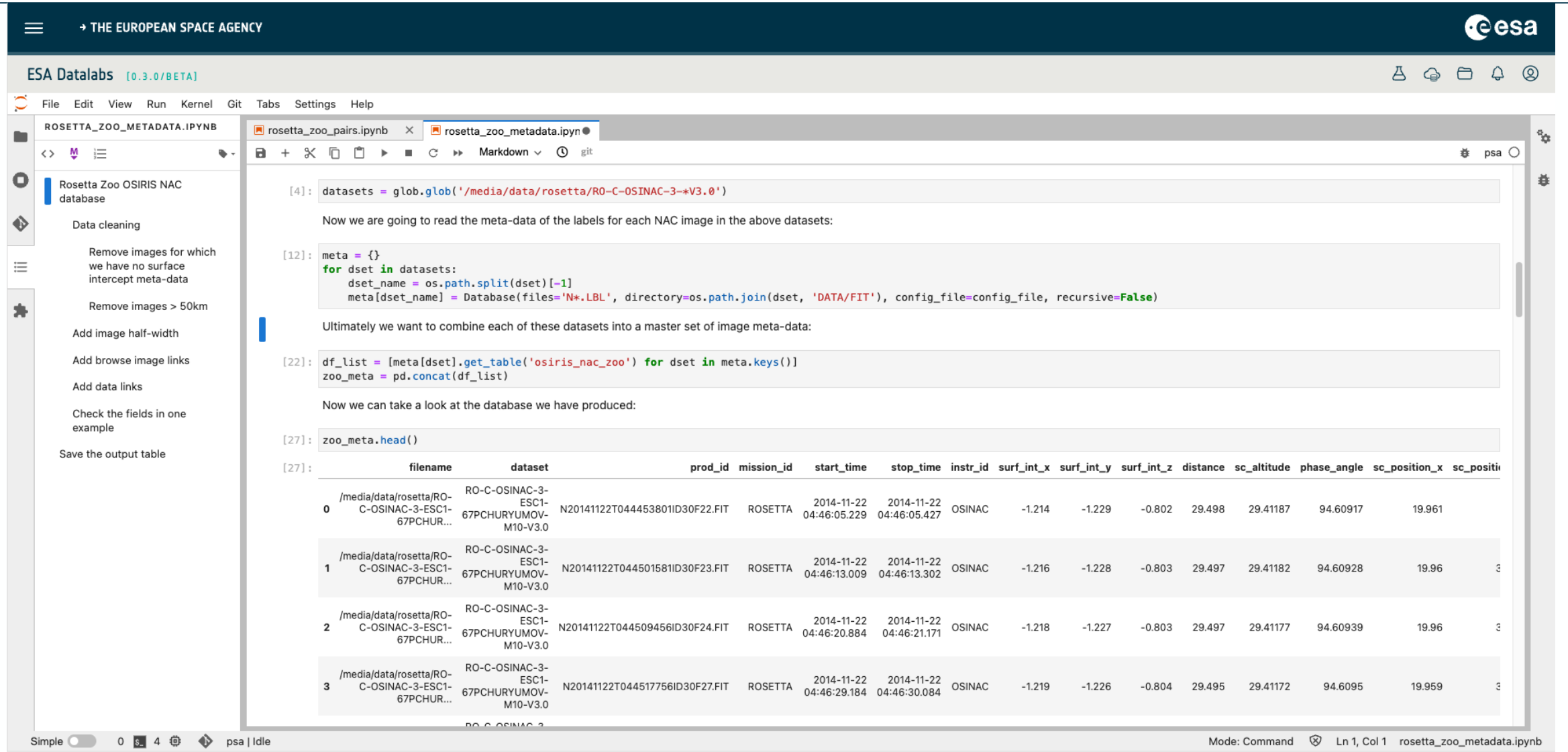
Examples: Rosetta/COSIMA



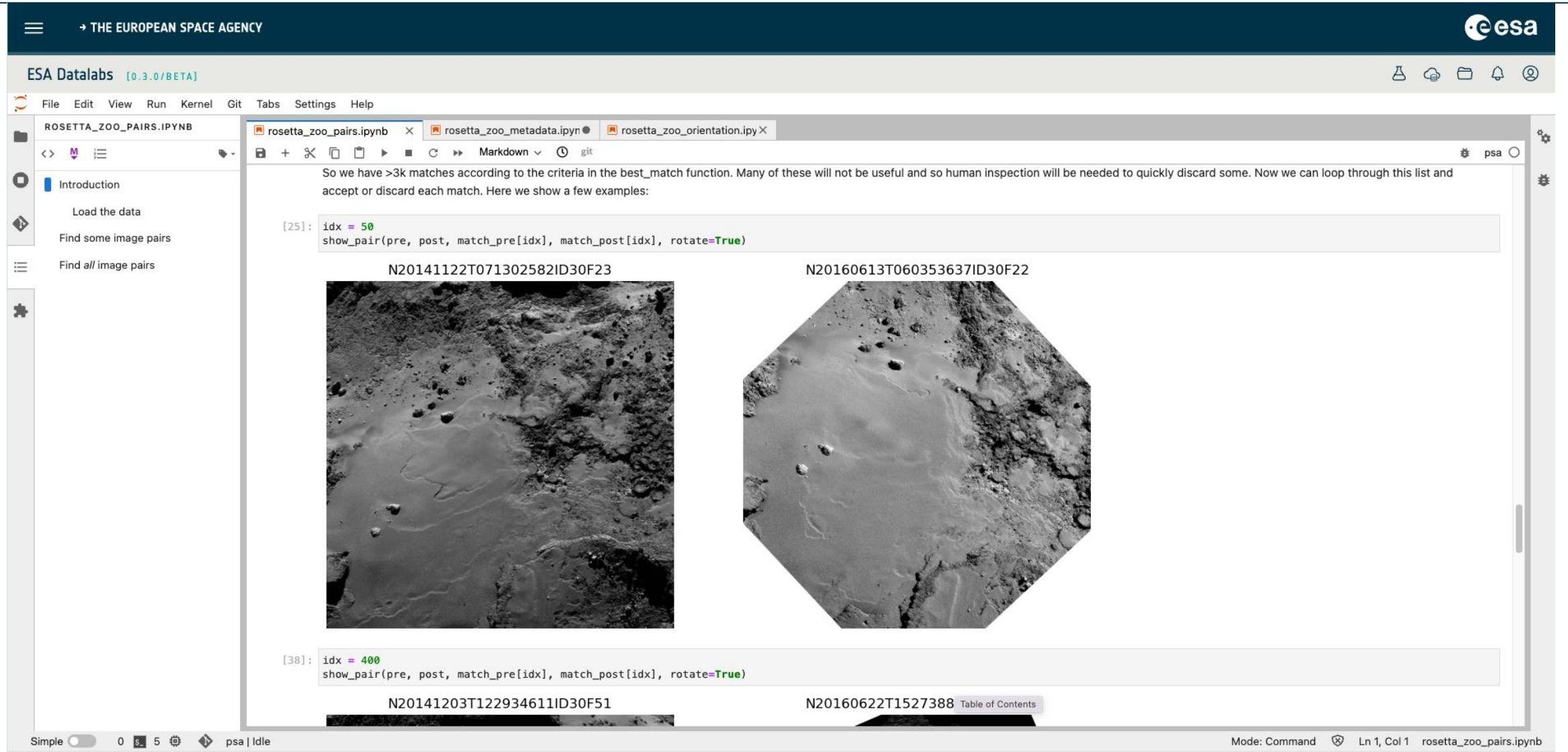
Examples: OSIRIS image matches

- Developed as part of the Zooniverse Rosetta Zoo project
  - Citizen science project to identify and classify changes on comet 67P using OSIRIS images
  - originally downloaded 20k meta-data label files from PSA
  - with DataLabs we don't have to do this!
- rosetta_zoo_metadata.ipynb (the real use of DataLabs is here!)
  - scrapes and cleans meta-data from matching data products and adds to pandas DataFrame
  - adds URLs to browse and data products
  - dumps to CSV
- rosetta_zoo_orientation.ipynb (not shown)
  - uses SPICE to add a north vector to each image to aid orientation
  - dumps to CSV
- rosetta_zoo_pairs (the final product)
  - defines a "matching function" to identify images of similar areas of comet 67P
  - displays candidate pairs (with optional image rotation)

Examples: OSIRIS image matches



Examples: OSIRIS image matches



Next steps

- PSA DataLab
  - request DataLab (once static mountpoints can be defined with a Lab)
  - update TAP with datalabs_path column
  - investigate how to provide authenticated access to private data
    - we are currently implementing an SFTP service which offers this

- Data tutorials
  - tidy up Rosetta notebooks and ask instrument teams to review, if possible
  - create new notebooks for other missions and instruments
    - BepiColombo – data tutorials are a WP in ongoing EXPRO contracts with the teams
    - other missions – developed in house, by interns, or archive scientists?

Issues to figure out

- Tutorials
  - how do we validate these, especially when instrument teams are no longer around?
    - reproducing results of a paper would be one way, reproducing calibration/processing steps would be another
  - how do we maintain the notebooks/tutorials
    - in the face of library updates, API changes, new data deliveries etc.
    - we can freeze our environment, but we want to stay more-or-less current
- Jupyter @ DataLabs in general
  - what is the typical user workflow?
    - often – develop in notebook, refactor into a module, re-write notebook to use module
    - do we offer any Git integration or similar? or rely on GitHub and co?
  - can we expose running notebooks externally so IDEs can hook into them?
    - i.e. connect to the running kernel (current URIs are localhost and probably behind a proxy)
  - can users manage Lab extensions themselves?
  - can users update packages in the base environment themselves?
    - in general how often do we update the Jupyter base environments