REPEATING PATTERNS IN THE DAY-TO-DAY WORK WITH PLANETARY MISSION DATA. K.-M. Aye[1],[1]Laboratory for Atmosphere and Space Physics, University of Colorado, Boulder (michael.aye@lasp.colorado.edu)

Introduction: After several years of processing and analyzing data from solar system missions, I have recognized a few repeating patterns in the supporting software I created for working with these data. Especially for the case of explorative data analysis, where one often has to look at and compare data from different quality levels, production runs, etc. two support structures turned out to be the most helpful patterns.

These most prevalent patterns are what I call path managers and data (or database) managers, and metadata searches.

Data(base) Manager: Often one has to work with different sources of data for the same analysis, either because one set is newer and not published at the same source, or because one set has been produced as a derivative from the other. In both cases, it is very helpful to define a software tool (a Python class in my case) that helps with easy finding and access to different databases. The helpful functionalities are: 1) finding the most recent files in a folder, in case subsequent processing is happening inside the same folder; 2) node/machine-dependent storage locations (because laptops usually have less capacity than desktops); 3), sub-dataset retriever functions that are able to filter the accessible data for keywords and/or ranges of parameters, easily providing a subset for the ongoing work task.

Path Manager: Each project will have, either defined by others, or by the researcher themselves, a structure of paths where different levels of data, or metadata data for the actual data are stored. For spacecraft missions this results in a large list of possible paths for each and every data product and it is very time-consuming to manually keep track of these while performing real-time explorative data analysis.

In recent projects I have implemented PathManagers to deal with this problem. Basically, a (currently hard-coded) class structure is defined that has attributes for each kind of sub-data paths underneath a common observation id. For example, for a HiRISE observation ID, there are paths to many products related to one observation ID, like COLOR, RED, and combined mosaics. During explorative analysis, I am able to call up a PathManager object for a given obsid and database, and an attribute called "red_mosaic" would provide the full path to that image product simply like so: "pm = PathManager(obsid, db); print(pm.red_mosaic)".

Metadata searches: Finding data often involves the search through metadata. To enbable this without using often cumbersome and mostly un-controllable web interfaces, the Planetary Data System archive requires the delivery of a cumulative index file that summarizes a chosen set of metadata for each observation id of a data set. I have found a way to ingest these metadata tables into a pandas DataFrame for convenient search queries. However, one current problem is that each delivery to any of the PDS nodes comes with a new cumulative index file that is stored into a new subfolder. This creates problems for tool creators like me that want to provide the newest metadata for a given mission instrument. It basically would create the need to parse the html code for the most recent folder that can be found which could be done at a hacking session at this conference, but recently, I have made progress in pleading with PDS nodes to create static URLs to these cumulative index files. The Rings-Moons node has agreed to create these static URLs that will link to the most recent delivered cumulative index file.

Abstractization: For my efforts in creating a toolset for planetary science in Python, similar to what "astropy" did for astrophysics, I believe it would be helpful to abstract these patterns into either a) configurable templates that should be easy adaptable for any new projects, so that it could become either part of the `planetpy` package, or b), become part of a so called cookiecutter project template that could be used to create a new software package for a new science project. I will provide concrete suggestions on how this could be done and am asking for community feedback on the best way forward.

The above mentioned metadata approach using pandas Dataframes is ready to be implemented into the `planetpy` project, it might just be more or less cumbersome to add the most recent cumulative index file to the current project, depending on the willingness of the PDS node to provide static links to these files.