HOW WE'LL KNOW WE CAN READ ALL THE DATA IN THE PDS: A TESTING METHODOLOGY FOR THE PLANETARY DATA READER!!! S. V. Kaufman[1], C. C. Million[2], M. A. St. Clair[2], [1]Million Concepts (sierra@millionconcepts.com) [2]Million Concepts

Introduction: The Planetary Data Reader (*pdr*) is an open-source Python application and library for reading planetary scientific data, particularly data archived in the Planetary Data System (PDS). We intend *pdr* to read all data products held by the PDS that comply with either version 3 or 4 of the PDS Data Standards (PDS3 or PDS4). Here we describe the current state of *pdr*, concentrating specifically on the methodology we've developed for determining dataset compatibility with our toolkit.

Installation. pdr is easy to install. We recommend the use of a conda environment. Regularly-updated installation instructions can be found here [1].

Workflow. pdr has a simple workflow that is immediately accessible to any user with basic knowledge of Python. You simply `import pdr` and call the function `pdr.read(fn)`, where `fn` is a data or label file of your choice; *pdr* returns all of the product's data and metadata in a Python object.

```
import pdr
data = pdr.read("frt00003e12_07_if166s_trr3.lbl")
data.show(band_ix=60)
```

Figure 1. Example workflow for *pdr* with a CRISM image [2] showcasing the .show convenience method, which allows users to easily view imaging data.

User Feedback and Data Requests. While *pdr* will be able to read all of the data in the PDS by the end of our primary development cycle (anticipated Q3 2024), we want this toolkit to be useful for the community well before then. We are prioritizing support for datasets based on our assessment of community need. If there's a data set you'd like us to prioritize support for, please make a request here [3]. If you have bug reports or other types of feature requests, please open a GitHub issue [4].

Cataloging the PDS: In order to scope and organize our compatibility testing process, we needed a lot of information about PDS holdings – data volume, file formats, numbers of files, volume structure, etc. There is no central catalog of the contents of the PDS that provides this information. We wrote bespoke multithreaded software to individually spider each PDS node and auxiliary mission node.

First, our spiders recursively navigated node directory link structures and recorded all URLs they encountered that appeared to point to data or metadata files. They also recorded all the directories they encountered, including any they attempted to traverse but failed. (In several cases, this led to the discovery of corrupted directories, which we brought to the attention of their housing nodes, who repaired them.)

Second, we used these URL lists to query file sizes by requesting the HTTP header (metadata) of each URL on the list created in the first step. For files whose headers did not include sizes, we parsed size information from the text of the parent web page. Concatenating these results gave us a database of names, URLs, and sizes for all discoverable files.

This database is a snapshot of the PDS at the time of spidering and will require periodic updates due to the release of new data (e.g., the Perseverance mission's first data release postdated our spidering effort). However, since individual missions are expected to have consistent file formatting standards, and all new missions will be subject to the PDS4 data standards [6] (which are strict compared to PDS3 [7]), keeping a continuously-updated list of files from ongoing missions is not critical.

Selecting Testing Subsets: Verifying *pdr*-produced outputs for every single file in the PDS would be an impossibly large task. Fortunately, most planetary data sets are reasonably well-organized and homogenous, so we can instead select a representative sampling of each "product type" in the PDS. By "product type," we mean a grouping of files that are similar to one another in both a science-domain sense and a data format sense; or, more pragmatically, files that, if *pdr* is able to open one, our expectation that *pdr* can open all the others greatly increases. (We are not attempting to produce a complete taxonomy of planetary data.) Factors influencing our identification of product type include (examples in parentheses): file formats (FITS, CSV), file structures (fields, dimensionality, data types), data structure categories (array/image, table), metadata structure (label fields, namespaces), level of

reduction (L0, RDR), explicit identification as a type by data providers, provenance (mission, instrument, software pipeline, producing institution), position in physical archive structure, target (the Moon, sky), and observational intent (calibration, science).

Practically speaking, we generally start by consulting a data set's software interface specification (SIS) or other primary documentation to see how the data providers categorized their products. We then supplement and subdivide this categorization (if necessary) by manually inspecting the archive structure and data. For example, the LROC SIS [5] refers to LROC NAC CDR files as a distinct kind of product. However, they come in two varieties: scaled I/F and radiance. We expect that these may work differently in *pdr* because their units and data types are different, so we treat them as two distinct product types in order to more easily find glitches and analyze unexpected outputs.

We then download one or two examples of each product type, manually open them using *pdr*, and record any modifications necessary to make *pdr* correctly read them. We then incorporate these modifications (if any) into *pdr*. At this point, we consider the data set "notionally supported."

Testing a Dataset: We then create a set of "rules" to identify download URLs for each of these product types – these might include a specific folder in the download link or a naming convention in the filename. We use these "rules" to filter our database of PDS files and create a list of all files of this product type. We then randomly select and download 200 of these files – or all the files, if there are fewer than 200 (for instance, LROC RDR products with larger swath coverage).

We open these files with *pdr* and hash their outputs using the MD5 algorithm. We also create browse versions of all data and metadata objects associated with the product, generally including at least an image or table and a parsed label text file, and possibly including many files. If there are one or more images, we produce unscaled and scaled versions of each image; we perform scaling based on scale and offset values specified in the label, common implicit special constant values, and special constant values specified in the label.

We manually inspect these browse products (comparing them, when possible, to other existing representations of the data) to ensure *pdr* read the product correctly. If there are unexpected outputs, we determine the source of the discrepancy, debug or modify it, and re-test the product set until the output is correct.

Once we are satisfied that *pdr* can read all files in the set correctly, we pare the set down to one or two files for each product type, which we then collate into a small list of "representative" files for the instrument or data set. We add this to a permanent test corpus for regression-testing modifications to *pdr*. After incorporation into the permanent test corpus, we consider a dataset "officially supported". A growing list of these data sets can be found at [8].

Regression Testing: Ongoing modifications to *pdr*'s code will be necessary for various reasons: adding new features, expanding compatibility (including compatibility with data sets that do not yet exist), code refactoring and "cleaning," and ensuring compatibility with new versions of software dependencies. When we update *pdr*'s code, we use our test suite to open our permanent test set, regenerate hashes for all products, and compare hash sets to ensure our code changes did not have unintentional consequences for officially supported data sets.

Hash comparison is a strict metric and can fail for reasons that do not affect the actual correctness of *pdr*'s output (operating environment changes, changes in software dependencies, etc.). We have already encountered this situation and are well-prepared to handle it; tools including versioned records of *pdr* and testing environments that preserve older versions of our dependencies can help us determine whether an apparent failure is, in fact, benign.

Acknowledgments: The development of *pdr* is supported by NASA grant No. 80NSSC21K0885. We would like to thank the Planetary Data System (PDS) for their continued cooperation with this project and for letting us stress their servers to index and grab very large amounts of data.

References:

[1] Installation instructions: https://github.com/MillionConcepts/pdr#readme

[2] Murchie, S., MRO CRISM TRDR, MRO-M-CRISM-3-RDR-TARGETED-V1.0, NASA Planetary Data System, 2006.

[3] Dataset request form: https://docs.google.com/forms/d/1JHyMDzC9LlXY4MOMcHqV5fbseSB096_PsLshAMqMWBw/edit

[4] GitHub issues: https://github.com/MillionConcepts/pdr/issues

[5] ASU. LROC EDR/CDR Data Product Software Interface Specification Version 1.18 (June 2010)

[6] Planetary Data System. Planetary Data System Standards Reference Version 1.17.0 (October 2021)

[7] Planetary Data System. Planetary Data System Standards Reference Version 3.8 (February 2009)

[8] List of supported datasets: https://github.com/MillionConcepts/pdr/blob/master/supported_datasets.md