




1 PDR: The Planetary Data Reader

2 Sierra Brown ¹, Michael St. Clair ¹, Chase Million ¹, Sabrina A.
3 Curtis ¹, K.-Michael Aye ², and Zack Weinberg ¹

4 1 Million Concepts LLC 2 Institut für Geologische Wissenschaften, Freie Universität Berlin

DOI: [10.xxxxx/draft](https://doi.org/10.xxxxx/draft)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Dan Foreman-Mackey](#) 

Reviewers:

- [@AndrewAnnex](#)
- [@gaelccc](#)

Submitted: 23 July 2024

Published: unpublished

License

Authors of papers retain copyright
and release the work under a
Creative Commons Attribution 4.0
International License ([CC BY 4.0](#))

5 Summary

6 The Planetary Data Reader, `pdr`, is an open-source Python-language package that reads data
7 stored in planetary science formats and converts it into standard Python objects. It typically
8 loads images as `ndarrays`, tables as `Pandas DataFrames`, and metadata and ancillary data as
9 strings or `dicts`. `pdr`'s interface is designed to be maximally accessible to the introductory
10 Python user. To read a data file, a user must simply `import pdr`, then run `pdr.read(fn)`,
11 where `fn` is the data file or a detached 'label' (metadata) file associated with it. `pdr` will
12 immediately load the product's metadata, then lazily load data objects when referenced.

13 `pdr` reads data products held by the Planetary Data System (PDS) that follow either PDS3
14 ([Jet Propulsion Laboratory, 2009](#)) or PDS4 ([Jet Propulsion Laboratory, 2024](#)) standards –
15 meaning in practice that they have metadata labels that generally follow one of these two
16 formats. (It also supports some common scientific interchange data formats that are not
17 PDS, including FITS.) We knew from the outset that many products in the PDS were not
18 fully standards-compliant, particularly those archived under PDS3; its holdings are extremely
19 diverse and span over half a century. For this reason, we took a **data-driven development**
20 approach rather than attempting to implement these standards to the letter. What this means
21 in practice is that we built `pdr` around a core of extremely flexible heuristics that permit it to
22 permissively accept and correctly handle many products that deviate from the standards.

23 We have developed these heuristics through an iterative design process centered on examination
24 of actually-existing data. We created manifests of the holdings of each of the PDS nodes,
25 then used them to help identify 'types' of data and retrieve representative samples of each
26 'type' (the PDS holds hundreds of millions of files with a total data volume in the petabytes,
27 so examining every file is impractical).

28 When we examine a new type, we verify the correctness of `pdr`'s behavior across our sample of
29 that type and make changes to `pdr` as necessary to support that type's characteristics. We
30 then add one or two individual products of that type to a data corpus we use for regression
31 testing. This has permitted us to design software that conforms to planetary data rather than
32 planetary data standards. Our methods for adding dataset support are further described in
33 Kaufman et al. (2022), and our testing toolchain can be found in Curtis et al. (2024).

34 `pdr` is an affiliate package of `planetarypy` ([PlanetaryPy Technical Committee, 2024](#)). It is
35 available on the Python Package Index and `conda-forge`.

36 Statement of need

37 *Just accessing data* is a major pain point for planetary scientists. Data archived under the
38 PDS3 standards can be especially challenging due to inconsistent, specialized, or flatly incorrect
39 formatting. While the newer PDS4 standards are significantly stricter, many of the holdings of
40 the PDS have not yet been migrated to this standard. `pdr` can remove months of preparatory
41 work, making it faster for scientists to get to core research tasks and making it much more

42 practical for them to incorporate data sets they haven't worked with previously into their
43 research.

44 The simplicity and consistency of pdr's API, along with its speed and stability, make it ideal
45 for use in automated data processing pipelines. pdr is currently used in a wide variety of
46 planetary projects. These include the Perseverance rover's Mastcam-Z tactical pipeline (St.
47 Clair, Million, et al., 2023) and PDS3 to PDS4 migration pipelines for data from Clementine
48 (St. Clair et al., 2021), Chandrayaan-1 M3 (Pieters et al., 2021), and the Viking Orbiter
49 cameras (St. Clair, Brown, et al., 2023). Its fast metadata parsing features make it especially
50 appealing for converting metadata standards across tens of millions of products. For example,
51 pdr is able to parse metadata and load the image arrays for 100 nominal-sized Mastcam-Z IOFs
52 using their attached PDS3 labels in 1.5 seconds and in 0.75 seconds to parse the metadata
53 alone. More complex labels such as the New Horizons ALICE calibrated Jupyter PDS3 labels
54 took pdr 1.7 seconds to parse metadata for 100 files.

55 Other packages

56 There is a very wide variety of software intended to read data in planetary science formats.
57 pdr's most important distinctions are its emphasis on breadth, simplicity, and high compatibility
58 with other tools. pdr incorporates some of this software, including pds4_tools (Small Bodies
59 Node & Nagdimunov, 2021) and astropy.io.fits (Astropy Developers, 2024). pdr uses these
60 packages to read, respectively, PDS4 and FITS files, converting their outputs into standard
61 Python objects to provide users with a common interface regardless of file format.

62 It is important to note that many pieces of software with narrower *format* scope than pdr
63 have wider *application* scope. For instance, GDAL (Rouault et al., 2024) and rasterio
64 (Mapbox, 2024) (which uses GDAL) read a narrower range of data and do not provide as
65 consistent or straightforward an interface, but will deal with map projection transformations;
66 plio (USGS Astrogeology, 2024) only reads data in a few formats, but is capable of applying
67 instrument-specific metadata-parsing rules. Many of these tools also offer write capabilities,
68 which pdr does not. Users who require write capabilities or subdomain-specific behaviors might
69 find narrowly-focused tools more appropriate; they might also find pdr useful as a preprocessor
70 for such tools.

71 Acknowledgements

72 The development of pdr is supported by NASA grant No. 80NSSC21K0885. We would like to
73 thank the Planetary Data System (PDS) for their continued cooperation with this project.

74 References

- 75 Astropy Developers. (2024). The Astropy Project. In *Github repository*. Github. <https://github.com/astropy/astropy>
- 76
- 77 Curtis, S. A., Brown, S. V., & St. Clair, M. A. (2024). Pdr-tests: The testing suite/toolchain
78 for 'pdr'. In *Github repository*. Github. <https://github.com/MillionConcepts/pdr-tests>
- 79 Jet Propulsion Laboratory. (2009). *Planetary Data System Standards Reference Version 3.8*.
80 nasa. https://pds.nasa.gov/datastandards/pds3/standards/sr/StdRef_20090227_v3.8.pdf
- 81 Jet Propulsion Laboratory. (2024). *Planetary Data System Standards Reference Version 1.22.0*.
82 nasa. <https://doi.org/10.17189/2ass-x557>
- 83 Kaufman, S. V., Million, C. C., & St. Clair, M. A. (2022). HOW WE'LL KNOW WE
84 CAN READ ALL THE DATA IN THE PDS: A TESTING METHODOLOGY FOR THE

- 85 PLANETARY DATA READER!!! *53rd Lunar and Planetary Science Conference*. <https://www.hou.usra.edu/meetings/lpsc2022/pdf/1119.pdf>
86
- 87 Mapbox. (2024). Rasterio. In *Github repository*. Github. <https://github.com/rasterio/rasterio/tree/main>
88
- 89 Pieters, C., Lunde, S., & Sunshine, J. (2021). *Chandrayaan-1 Orbiter Moon Mineralogy Mapper Collected Data Sets*. NASA PDS: USGS Imaging Node. <https://doi.org/10.17189/f8xf-6a29>
90
91
- 92 PlanetaryPy Technical Committee. (2024). *Planetarypy*. <https://planetarypy.org/>
- 93 Rouault, E., Warmerdam, F., Schwehr, K., Kiselev, A., Butler, H., Łoskot, M., Szekeres, T.,
94 Tourigny, E., Landa, M., Miara, I., Elliston, B., Chaitanya, K., Plesea, L., Morissette,
95 D., Jolma, A., Dawson, N., Baston, D., de Stigter, C., & Miura, H. (2024). GDAL:
96 Geospatial Data Abstraction Library. In *Github repository*. Github. <https://doi.org/https://zenodo.org/records/12545688>
97
- 98 Small Bodies Node, P., & Nagdimunov, L. (2021). PDS4 Tools. In *Github repository*. Github.
99 https://github.com/Small-Bodies-Node/pds4_tools
- 100 St. Clair, M. A., Brown, S., & Million, C. (2023). *Viking Orbiter Imaging Bundle*. NASA
101 PDS: USGS Imaging Node. <https://d1nxexkqx2p6yf.cloudfront.net/PDS4/bundle.xml>
- 102 St. Clair, M. A., Million, C. C., Brown, S. V., & Rice, M. S. (2023). Automated Spectral
103 Image Processing Techniques in the Marslab Family of Applications. *6th Planetary Data
104 Workshop*. <https://www.hou.usra.edu/meetings/planetdata2023/pdf/7050.pdf>
- 105 St. Clair, M. A., Million, C., & Ianno, A. (2021). *Clementine Imaging Bundle*. NASA PDS:
106 USGS Imaging Node. <https://doi.org/10.17189/07q9-ph18>
- 107 USGS Astrogeology. (2024). *pilo: Planetary Input/Output*. In *Github repository*. Github.
108 <https://github.com/DOI-USGS/plio>