**UPDATE ON PYSAT: A SPECTRAL DATA ANALYSIS TOOL FOR PLANETARY SCIENCE.** L.R. Gaddis[1], J. Laura[1], R.B. Anderson[1], T. Hare[1], R. Klima[2], F. Morgan[2], C. Viviano-Beck[2], N. Finch[3], A. Paquette[3], K. Rodriguez[3], A. Sanders[3]. [1]U.S. Geological Survey, Astrogeology Science Center, Flagstaff, AZ; [2]Applied Physics Laboratory, Johns Hopkins University, Laurel, MD; [3]Northern Arizona University, Flagstaff, AZ. (lgaddis@usgs.gov).

**Introduction:** The Python Spectral Analysis Tool (PySAT) is a software library designed to enable visualization, thematic image derivation, and spectral analysis of planetary spectral data in a cross-platform, open-source environment. Major aims of the PySAT project are to deliver complex processing algorithms for creating high-level thematic image products from archived PDS data, to make it simple for algorithms to be modified or added, and to free planetary scientists from the need to use expensive or closed-source software. Recent PySAT development has focused on the extension and solidification of an Application Program Interface (API) for orbital and ground based spectrometers. This includes the development of Pandas-based functional access to spectral data and associated metadata, the solidification of the library internals, and the development of Jupyter notebooks to demonstrate use of the API. In the current form, the high level API exposed by the library is ideal for the scientist seeking to perform exploratory spectral data analysis, prototype algorithms without the need to manage the underlying spectral data, and generate rapid visualizations to support analysis. All PySAT development is public facing and freely available via https://github.com/USGS-Astrogeology/PySAT. All development described here is occurring within the 'dev' branch.

**Background:** Spectral data ingestion, spectral analysis and product generation are supported by PySAT through the implementation of functions such as continuum correction, noise removal, band depth, position and shape analysis and derivation of related thematic products. The PySAT library is not coupled to a GUI and can readily be integrated into data processing workflows. The PySAT library is developed in Python and integrated into the Python scientific computing stack. Python (https://www.python.org/) exemplifies rapid code development, an iterative workflow, and source readability. PySAT code is 3.x compliant because broad Python 2.x support is slated for deprecation by the core Python team. Use of Python provides a rapid, interactive development environment that does not require code compiling. This environment fosters a simple workflow (*e.g.*, implement, test, refactor, utilize) that facilitates efficient code prototyping and delivery. For example, code can be rapidly developed and delivered with loose requirement specifications, then iteratively refactored with input from science users as needs are refined.

Several Python libraries are used in PySAT: (1) ***NumPy*** (Numerical Python) and (2) ***SciPy*** (Scientific Python), which provide a suite of numerical analysis tools, access to the robust LaPack and Blas libraries for linear algebra operations, as well as a range of additional mathematical functions, (3) ***Pandas*** for R style dataframe representation and exceptional processing performance, (4) ***GDAL*** (Geospatial Data Abstraction Library) which provides access to over 120 image data formats, including PDS3, ISIS2/3, FITS, ENVI, JPEG 2000, PNG and TIFF [5], (5) ***Matplotlib*** for data visualization, and (6) ***scikit-learn*** for machine learning and multivariate analysis algorithms. Together these libraries form a highly effective processing suite for scientific data exploration, visualization, and analysis. These libraries, except for GDAL, PySide, and scikit-learn, are core components of the Python Scientific computing stack and widely available on all platforms. The freely available ***Anaconda Python*** software distribution mechanism simplifies installation of dependencies.

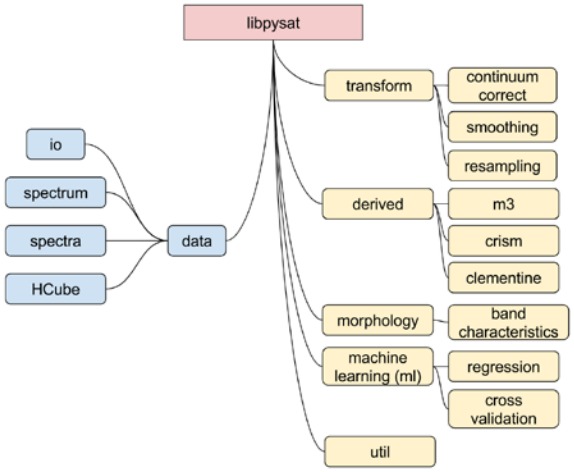


***Figure 1****: PySAT library architecture; data structures at left, analytical functions at right. The root, libpysat, differentiates this library from the PySAT GUI.*

**PySAT Library:** The PySAT library (***Figure 1***) is a set of modules with a high level API to support data ingestion, exploratory spectral data analysis, the creation of processing workflows, and the development of stand-alone applications. The core data structures of the PySAT library are built around Pandas dataframes for three reasons. First, our extended Pandas data frame objects support both positional and label-based spectral data analysis with user defined precision tolerances. In practice, this means that accessing specific wavelengths can be done with truncated notation that need not be de-

fined to floating point precision. Second, Pandas natively supports SQL-style queries and a wide array of statistical analysis operations without the need for additional implementation (e.g., ***Figure 2***). For example, the computation of descriptive statistics for an individual spectrum or region of interest can be performed natively without the need for project developers to write or maintain code. Finally, Pandas integrates natively with MatPlotLib, offering off-the-shelf visualizations. In terms of functionality, this library provides analytical capabilities to perform basic processing tasks across multi- and hyper-spectral instruments as well as specific functionality designed for the NASA Moon Mineralogy Mapper ($M^3$) on the ISRO Chandrayaan-1 mission and the NASA Compact Reconnaissance Imaging Spectrometer for Mars (CRISM) from NASA's Mars Reconnaissance Orbiter instruments. The $M^3$ thematic product algorithms library consists of ~50 algorithms [5] and the CRISM thematic product library has ~60 algorithms [6]. In updating the library, we have currently implemented and exposed the $M^3$ derived product algorithms. At the time of writing these algorithms are available in a beta form and still require validation by members of our science team. Finally, the library supports the analysis of point spectrometer data with an emphasis on preprocessing and multivariate analysis of Laser-Induced Breakdown Spectroscopy (LIBS) spectra [3, 4] in conjunction with the PySAT-GUI application.

**Graphical User Interface (GUI)**: We plan to develop a GUI within the QGIS application to provide a subset of the total PySAT functionality in a user-friendly (non-developer or computational scientist) form. QGIS is an industry-standard, open-source GIS with a wide user base, planetary data and projection support, a robust development community, and works across all modern platforms (Macintosh, Windows, Linux, etc.). QGIS uses multi-threaded image rendering, allowing us to leverage high performance GUI infrastructure for large planetary data products. For example, we leverage the built-in QGIS support for Open Geospatial Consortium (OGC) existing live mapping layers (see Astrogeology WMS Map Layers here: https://astrowebmaps.wr.usgs.gov/webmapatlas/Layers/maps.html) for use as a base image or spatial context for spectral data analysis and visualization [9]. We are awaiting the release of the new QGIS version with Python 3.x support and the ability to leverage Qt WebViews. These views will allow the development of Javascript-based front-ends that can be easily ported to traditional web clients. Future extensibility is supported for use of spectral modeling tools such as the Modified Gaussian Method (MGM; [10]), the functionality to integrate and compare spectra with spectral libraries such as those from the NASA/Brown University Reflectance Laboratory (RELAB, [11, 12]; see http://www.planetary.brown.edu/relab/), or any other spectral data requiring visualization and analysis.

```
# Access indices 1,2,3,5,8,13 in the spectrum object
ref = s.loc[['REF1', 'REF2'], :, [512, 518, 542]].sort_index(level=1)
ref.head()
```

| | | 512.6 | 518.4 | 542.8000000000001 |
|---|---|---|---|---|
| minor | id | | | |
| REF1 | 0 | 0.0402 | 0.0487 | 0.0551 |
| REF2 | 0 | 0.0397 | 0.0482 | 0.0545 |
| REF1 | 1 | 0.0414 | 0.0502 | 0.0568 |
| REF2 | 1 | 0.0409 | 0.0497 | 0.0562 |
| REF1 | 2 | 0.0391 | 0.0474 | 0.0537 |

***Figure 2:*** *Sample API usage in a Jupyter Notebook using Kaguya Spectral Profiler data and label-based wavelength indexing.*

**Conclusion:** Prototype versions of both PySAT GUIs are available on GitHub: The orbital data version and the library for spectral extraction, manipulation, and visualization is online (http://github.com/USGS-Astrogeology/pysat) and the version for visualization and analysis of LIBS and other point spectral data also is available (see https://github.com/USGS-Astrogeology/PySAT_Point_Spectra_GUI). Development and implementation of orbital derived product creation is underway and will be tested and validated by this research team before release (expected in late 2018).

**References:** [1] Gaddis, L. et al. (2017) 48th LPSC, abs. #2548. [2] Gaddis et al. (2017) 3rd Planetary Data Workshop, abs. #.7060 [3] Anderson, R. et al. (2017) AGU Fall Meeting, paper #233006. [4] Anderson, R. et al., this volume. [5] Moriarty, D. et al. (2013) *JGR-P* 118, 2310-2322. [6] Viviano-Beck, C. et al. (2014) *JGR-P* 119(6), 1403-1431. [7] Sides, S. et al. (2017) *LPS XLVIII*, 48th LPSC, abs. #2739. [8] Mitchell, T. et al. (2013) *Geospatial Power Tools*, 338 p. [9] Hare, T. et al. (2007) *LPS XXXVII*, abs. #2364. [10] Sunshine, J. et al. (1990) *JGR* 95, 6955-6966. [11] Pieters, C. and Hiroi, T. (2004) *LPS XXXV*, abs. #1720. [12] Milliken, R. et al. (2016) *LPS XLVII*, abs. #2058.