PYHAT 2019 UPDATE: THE PYTHON HYPERSPECTRAL ANALYSIS TOOLS FOR PLANETARY SCIENCE (FORMERLY KNOWN AS PYSAT). L.R. Gaddis[1], J. Laura[1], T. Hare[1], E. Gault[2], A. Paquette[2], T. Thatcher[2]. [1]U.S. Geological Survey, Astrogeology Science Center, Flagstaff, AZ; [2]Northern Arizona University, Flagstaff, AZ. (lgaddis@usgs.gov).

Introduction: The Python Hyperspectral Analysis Tools (PyHAT), formerly called Python Spectral Analysis Tools (PySAT), was renamed because of a naming conflict with the acronym of another Python toolkit. Goals of the PyHAT 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 [1-3]. PyHAT 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. PyHAT development has focused on development of an Application Program Interface (API) for orbital and ground-based [4] spectrometers. This includes the development of Pandas-based functional access to spectral data, a library of tools for spectral processing and analysis (with particular application to NASA's Moon Mineralogy Mapper (M^3) and Compact Reconnaissance Imaging Spectrometer for Mars (CRISM) datasets) and the use of Jupyter notebooks to validate the API and tools using planetary hyperspectral data. The API exposed by the library supports exploratory spectral data analysis, the ability to view and modify algorithms easily, quick visualization of plots and results, and saving them in common data formats. PyHAT is available on GitHub at https://github.com/USGS-Astrogeology/PyHAT.

Background: Spectral data ingestion, spectral analysis and product generation are supported by PyHAT through the use of functions such as continuum correction, noise removal, band depth, position and shape analysis and derivation of related thematic products. The PyHAT library is developed in Python 3.x and integrated into the Python scientific computing stack. Python (https://www.python.org/) exemplifies rapid code development, an iterative workflow, and source readability. Use of Python provides a rapid, interactive development environment that does not require code compiling. This environment fosters a simple workflow that facilitates efficient code prototyping and delivery. For example, code can be rapidly developed and delivered with loose requirement specifications, then refactored with input from science users as needed.

Several Python libraries are used in PyHAT: (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.

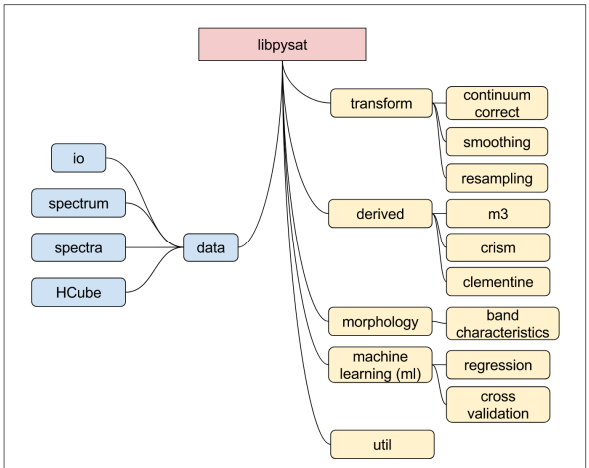

Figure 1*: PyHAT library architecture; data structures at left, analytical functions at right.*

PyHAT Library: The PyHAT 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 PyHAT 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 defined 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. 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.

The PyHAT library provides analytical capabilities to perform basic processing tasks across multi- and hyper-spectral instruments as well as specific functionality designed for M^3 and CRISM hyperspectral data (e.g., ***Figure 2***). The M^3 thematic product algorithms library consists of ~50 algorithms [5] and the CRISM thematic product library has ~60 algorithms [6]. We have implemented the M^3 and CRISM derived product algorithms and they are currently being validated. Additional validation by members of our team (R. Klima, C. Viviano-Beck, F. Morgan) is planned for later this year. Note that the PyHAT library also supports the analysis of point spectrometer data with an emphasis on preprocessing and multivariate analysis of Laser-Induced Breakdown Spectroscopy (LIBS) spectra [3, 4, 9] in conjunction with the PyHAT GUI application.

Graphical User Interface (GUI): We are currently working to develop a GUI within the QGIS application to provide a subset of the total PyHAT functionality in a user-friendly (non-developer or computational scientist) form. QGIS is an industry-standard, open-source Geographic Information System (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 [10]. Future PyHAT extensibility is supported for use of spectral modeling tools such as the Modified Gaussian Method (MGM; [11]), the functionality to integrate and compare spectra with spectral libraries such as those from the NASA/Brown University Reflectance Laboratory (RELAB, [12, 13]; see http://www.planetary.brown.edu/relab/), or any other spectral data requiring visualization and analysis.

Conclusion: Prototype versions of both PyHAT 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/PyHAT) and the version for visualization and analysis of LIBS [9] and other point spectral data also is available (see https://github.com/USGS-Astrogeology/PyHAT_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 2019).

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] Anderson et al., this meeting [10] Hare, T. et al. (2007) *LPS XXXVII*, abs. #2364. [11] Sunshine, J. et al. (1990) *JGR* 95, 6955-6966. [12] Pieters, C. and Hiroi, T. (2004) *LPS XXXV*, abs. #1720. [13] Milliken, R. et al. (2016) *LPS XLVII*, abs. #2058.

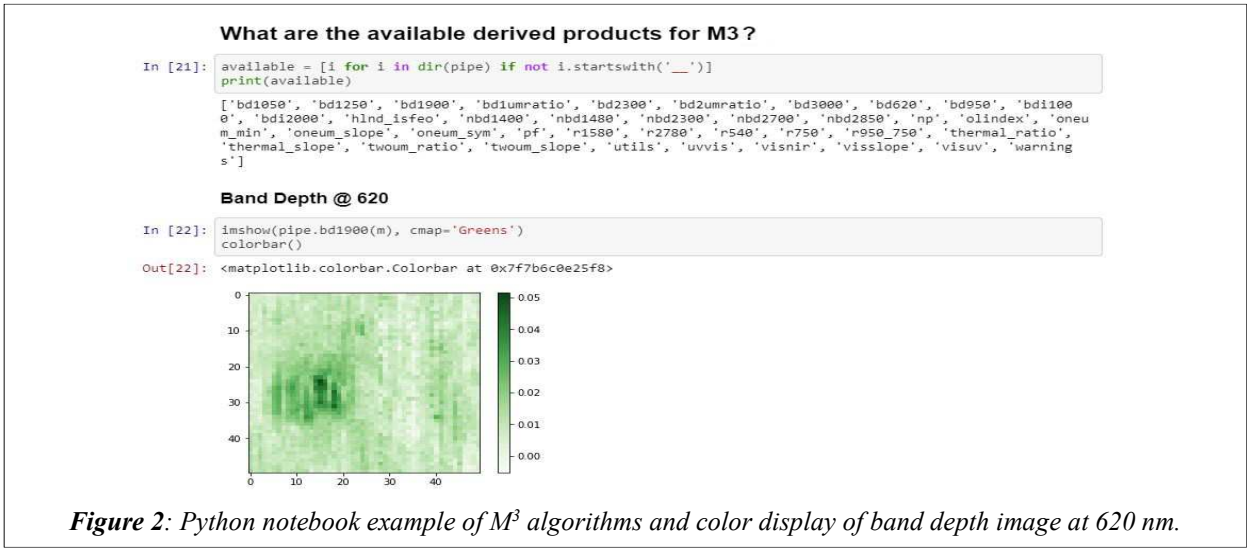

Figure 2*: Python notebook example of M^3 algorithms and color display of band depth image at 620 nm.*