THE PYTHON SPECTRAL ANALYSIS TOOL (PYSAT) FOR POWERFUL, FLEXIBLE, AND EASY PREPROCESSING AND MACHINE LEARNING WITH POINT SPECTRAL DATA

R.B. Anderson[1], N. Finch[1], S. Clegg[2], T. Graff[3], R.V. Morris[3], J. Laura[1]; [1]U.S. Geological Survey, Astrogeology Science Center, Flagstaff, AZ (rbanderson@usgs.gov); [2]Los Alamos National Laboratory, [3]NASA Johnson Space Center.

Introduction: Many planetary spectroscopy instruments such as ChemCam, SuperCam, Alpha Particle X-Ray Spectrometer (APXS), Mossbauer, Planetary Instrument for X-ray Lithochemistry (PIXL), Mini-TES, etc. collect point spectral data. Interpretation of these data is vital to the understanding of the geology of the targets and sites analyzed, but point spectral data are considerably more difficult to work with than images. Even for members of instrument teams, it can be challenging to apply new processing and analysis techniques to the data.

We have developed the free and open-source Python Spectral Analysis Tool (PySAT) library and point spectra interface to enable the planetary community to process and analyze point spectra without requiring programming expertise. This work is distinct from, but complimentary to, PySAT development geared toward orbital imaging spectrometer data [1].

Data Format: PySAT uses the Pandas library [2] to efficiently store spectra and associated metadata in a single data frame. The primary format for point spectral data is a comma-separated value (.csv) file with spectra and associated metadata stored in rows. The .csv file has two-level column labels with the top row indicating broad categories of data (e.g. wavelength, metadata, composition) and the second row indicating specific categories (e.g. '240.811', 'Target Name', 'SiO_2'). The PySAT point spectra tool can read individual PDS-format "clean, calibrated spectra" (CCS) files from ChemCam into the PySAT .csv format. We also provide laboratory data from a Laser-Induced Breakdown Spectroscopy (LIBS) instrument at Johnson Space Center in the proper format to be used with PySAT.

Capabilities:

Preprocessing: The PySAT point spectra tool provides a number of useful preprocessing capabilities. Once loaded, data frames can be manipulated by removing rows, splitting a single data frame, merging multiple data frames into one, multiplying all spectra in a data frame by a vector, or finding the derivative of each spectrum. Spectra in one data frame can also be interpolated onto the spectral channels from another data frame, a first step in combining data from different instruments.

A mask, specified by a simple .csv file, can be applied to all of the spectra in a data frame, and spectra can be normalized to the integrated signal within specified wavelength range(s). Spectra can also be grouped into a number of "folds" in preparation for model validation and testing, with the folds stratified on a single column of the data frame's metadata (for example, the SiO_2 content) to ensure a similar distribution of that variable in each fold. PySAT leverages the scikit-learn machine learning library [3] to enable several different dimensionality reduction methods: principal components analysis (PCA), two different independent component analysis (ICA) algorithms, as well as t-distributed stochastic neighbor embedding (t-SNE), and Locally Linear Embedding (LLE). The PySAT library also includes a number of continuum removal algorithms provided by [4]. We have also implemented several outlier removal methods, including Isolation Forest, Local Outlier Factor, One-class SVM, and Elliptic Envelope.

We have also added a "peak area" method which identifies local minima and maxima in the spectra, sums the signal between the minima, and saves the new value under the wavelength of the maximum. This effectively collects the signal from a peak into a single channel, shrinking the data set. This has been shown to improve calibration based on weak emission peaks [5]. Future updates will include true peak-fitting capabilities.

Regression: Regression methods can be used to generate models capable of converting observed spectra into predictions of target properties, such as chemical composition. PySAT leverages scikit-learn [3] to allow users to train a variety of regression models, including: Automatic Relevance Determination (ARD), Bayesian Ridge Regression (BRR), Elastic Net, Gaussian Process Regression, Least Angle Regression (LARS), least absolute shrinkage and selection operator (Lasso), Ordinary Least Squares (OLS), Orthogonal Matching Pursuit (OMP), Partial Least Squares (PLS), Ridge regression, and Support Vector Regression (SVR). To assist users in identifying the optimal parameters for these methods, a flexible cross validation option is available using the stratified folds defined in preprocessing.

Once a regression model has been trained, it can be used for prediction. If multiple models have been trained, they can be combined to implement submodel regression [6]. The current ChemCam calibration uses blended PLS submodels [7], but there is no requirement in PySAT that the same regression method be used for all submodels, providing added flexibility. The ranges over which the submodels are blended in

PySAT can optionally be optimized based on performance on training data.

Visualization: PySAT also includes Matplotlib-based [8] options for producing point and line plots, as well as plots of scores and loadings to visualize PCA and ICA results. Figure 1 shows some example plots.

Interface: We have also developed a graphical user interface based on PyQt5 [9] to make all of the above capabilities accessible for non-programmers. The graphical interface is based on the concept of "workflows", comprising individual processing steps or "modules" arranged in a user-specified order. The interface includes a progress bar to indicate when the program is running a calculation. As each module is run it is grayed-out, indicating progress through the workflow. A console view at the bottom of the interface prints messages as the modules run. The interface allows users to re-run or delete the last module, and stop the run partway through.

Once the user has determined the optimal order for the steps in the workflow, it can be saved and restored at a later date. This saving and restoration process uses Python's "Pickle" capability, and provides a convenient way to store and later replicate the exact processing steps involved in deriving results from a data set. For example, the methods and plotting commands used to produce the results and figures in a publication could be stored and included as supplemental information along with the manuscript.

Future Work: Both the PySAT point spectra tool GUI [10] and the underlying code containing the core functionality of PySAT for point spectra [11] are available on github. As development continues, several additional capabilities will be added, including: clustering and classification; calibration transfer; and interactive plotting. The course of development and prioritization of different features will also be guided by feedback from users. We will also work to fully document the tool and develop tutorials.

Although the tool was designed primarily based on analysis of LIBS spectra, it can be easily applied to any other spectral data. The tool provides a flexible and powerful framework enabling scientists to process and analyze spectral data using multivariate and machine learning methods, leading to improved scientific results.

References: [1] Gaddis et al. (2017) Planetary Data Workshop, #7060 [2] http://pandas.pydata.org/ [3] http://scikit-learn.org/ [4] Giguere, S., Carey, C.J., Boucher, T., et al. (2013) Proc. 5th IJCAI Workshop on Artificial Intelligence in Space. [5] Clegg, S.M. et al. (2017) LPSC XLVIII, #2439 [6] Anderson, R.B., et al. (2017) Spectrochim. Acta B, 129, 49–57. doi:http://dx.doi.org/10.1016/j.sab.2016.12.003 [7] Clegg, S. et al. (2017), Spectrochim. Acta B, 129, 64-85. http://doi.org/10.1016/j.sab.2016.12.003 [8] http://matplotlib.org/ [9] https://riverbankcomputing.com/software/pyqt/intro [10] https://github.com/USGS-Astrogeology/PySAT_Point_Spectra_GUI [11] https://github.com/USGS-Astrogeology/PySAT

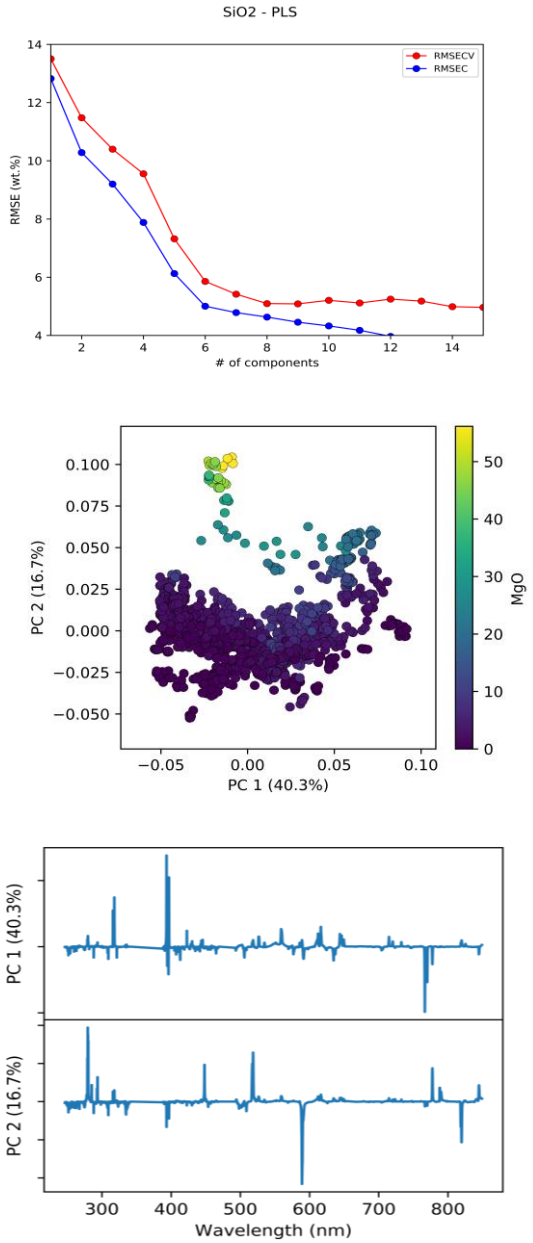

Figure 1: *Top*: Cross validation results for a PLS model of SiO2 based on LIBS spectra. *Middle*: Plot of the PCA scores for the first two principal components of a database of LIBS spectra. Points are color coded according to MgO content. *Bottom*: The corresponding PCA loadings for the first two components.