specdal Documentation

Release 2.0.0

Young Lee

May 02, 2018

Contents:

CHAPTER 1

Introduction

SpecDAL is a Python package for loading and manipulating field spectroscopy data. It currently supports readers for ASD, SVC, and PSR spectrometers. SpecDAL provides useful functions and command line scripts for processing and aggregating the data.

1.1 Interface

There are three options for using SpecDAL.

1. Python interface

   The lowest level interface is for users to import `specdal` as a Python module. Functions in `specdal` are written to operate directly on Pandas Series and DataFrames. `specdal` also provides classes that wrap around Pandas objects for convenience to users not familiar with Pandas.

   Users at this level are encouraged to check out the *data model*, Notebook examples , and the *API*.

2. Command line interface

   Alternatively, users can utilize the command line scripts that `specdal` provides. The following scripts are currently distributed:

   - *specdal_info*: displays key information in a spectral file
   - *specdal_pipeline*: converts a directory of spectral files into .csv files and figures

3. Graphical User Interface (GUI)

   At the highest level, `SpecDAL` provides a GUI that requires no programming. GUI can be handy for tasks such as outlier detection. GUI is provided as an executable, `specdal_gui` on Linux/Mac and `specdal_gui.exe` on Windows.

1.2 Examples

Check out the example Notebooks here.

CHAPTER 2

Installation

SpecDAL is available via pip (`pip install specdal`) or on Github. This page provides detailed walkthrough of the installation process intended for users who are not comfortable in Python environment.

2.1 Prerequisites

- python3
- pip3

2.1.1 Setting up the virtual environment (recommended)

Although not necessary, it is good practice to install Python packages in a virtual environment. Virtual environments provide an isolated and self-contained environment for your Python session, which can help prevent conflicts across packages. We will walk through the process of creating one on Ubuntu Linux for demonstration.

- Install virtualenv using pip installer.

```
$ pip install --user virtualenv
```

- Create a directory for storing virtual environments.

```
$ mkdir ~/venv
```

- Create a new virtual environment called `specdal_env` running python3 by default.

```
$ virtualenv -p python3 ~/venv/specdal_env
```

  If you're curious, you can navigate to that directory and find all the components that make up a Python environment. For example, packages are installed in `~/venv/specdal_env/lib` and binaries are stored in `~/venv/specdal_env/bin`.

- Before starting a Python session, we can activate the virtual environment as follows.

```
$ source ~/venv/specdal/bin/activate
```

Note: On windows, there should be an executable `~/venv/specdal/bin/activate.exe` with a similar effect.

You'll notice the name of your virtual environment in parentheses.

```
(specdal_env) $
```

- Once in this environment, we can install and use `SpecDAL` or other packages.

```
(specdal_env) $ ... # install specdal
(specdal_env) $ ... # write and run programs
```

- When we're done, we can exit the virtual environment.

```
$ deactivate
```

2.2 Install via pip

- Stable version

```
$ pip3 install specdal --upgrade
```

- Latest development version

```
$ pip3 install specdal --pre
```

2.3 Install from Github

SpecDAL can be found on Enspec's Github repo. Stable release can be found on `master` branch and the development version on `dev` branch.

2.3.1 Github walkthrough

1. Open terminal or Git-bash and navigate to the desired directory, `~/specdal` for this demo.

   `cd ~/specdal`

2. The following command will clone the SpecDAL's Github repository.

   ```
   $ git clone https://github.com/EnSpec/SpecDAL.git
   ```

   You'll notice a new subdirectory `SpecDAL` with the source code.

3. Install SpecDAL.

   ```
   $ cd ./SpecDAL
   $ python setup.py install
   ```

2.3.2 Install in development mode

If you'd like to modify SpecDAL's source, it's useful to install the package in development mode.

- Install in development mode

```
$ python setup.py develop
```

- Modify the source and run/test it.
- Uninstall development mode

```
$ python setup.py develop --uninstall
```

CHAPTER 3

Data Model

SpecDAL relies on Pandas data structures to represent spectroscopy measurements. A single measurement is stored in pandas.Series while a collection of measurements is stored in pandas.DataFrame. SpecDAL provides Spectrum and Collection classes that wraps Series and DataFrames along with spectral metadata. Spectral operators, such as interpolation, are provided as functions on pandas objects or as methods of specdal's classes.

3.1 Pandas Representation of Spectra

3.1.1 Series - single spectrum

3.1.2 DataFrame - collection of spectra

3.2 Spectrum and Collection Classes

3.2.1 Spectrum - single spectrum

3.2.2 Collection - collection of spectra

3.3 Operators

CHAPTER 4

API Reference

This is the class and function reference page of SpecDAL.

4.1 Spectrum

class specdal.containers.spectrum.**Spectrum**(*name=None*, *filepath=None*, *measurement=None*, *measure_type='pct_reflect'*, *metadata=None*, *interpolated=False*, *stitched=False*, *jump_corrected=False*, *verbose=False*)

Class that represents a single spectrum

name: string Name of the spectrum.

filepath: string (optional) Path to the file to read from.

measurement: pandas.Series Spectral measurement

metadata: OrderedDict Metadata associated with spectrum

Spectrum object stores a single spectral measurement using pandas.Series with index named: "wavelength".

get_pct_reflect(*dataframe*)
Helper function to calculate pct_reflect from other columns

pd.Series object for pct_reflect

interpolate(*spacing=1*, *method='slinear'*)

jump_correct(*splices*, *reference*, *method='additive'*)

read(*filepath*, *measure_type*, *verbose=False*)
Read measurement from a file.

stitch(*method='mean'*)

4.2 Collection

class specdal.containers.collection.**Collection**(*name*, *directory=None*, *spectra=None*, *measure_type='pct_reflect'*, *metadata=None*, *flags=None*)

Represents a dataset consisting of a collection of spectra

append(*spectrum*)
insert spectrum to the collection

data
Get measurements as a Pandas.DataFrame

data_with_meta(*data=True*, *fields=None*)
Get dataframe with additional columns for metadata fields

data: boolean whether to return the measurement data or not

fields: list names of metadata fields to include as columns. If None, all the metadata will be included.

pd.DataFrame: self.data with additional columns

flags
A dict of flags for each spectrum in the collection

groupby(*separator*, *indices*, *filler=None*)
Group the spectra using a separator pattern

OrderedDict consisting of specdal.Collection objects for each group key: group name value: collection object

interpolate(*spacing=1*, *method='slinear'*)

jump_correct(*splices*, *reference*, *method='additive'*)

max(*append=False*, *ignore_flagged=True*)

mean(*append=False*, *ignore_flagged=True*)

median(*append=False*, *ignore_flagged=True*)

min(*append=False*, *ignore_flagged=True*)

plot(**args*, ***kwargs*)

read(*directory, measure_type='pct_reflect', ext=['.asd', '.sed', '.sig', '.pico', '.light'], recursive=False, verbose=False*)
read all files in a path matching extension

spectra
A list of Spectrum objects in the collection

std(*append=False*, *ignore_flagged=True*)

stitch(*method='max'*)

to_csv(**args*, ***kwargs*)

specdal.containers.collection.**df_to_collection**(*df*, *name*, *measure_type='pct_reflect'*)
Create a collection from a pandas.DataFrame

df: pandas.DataFrame Must have spectrum.name as index and metadata or wavelengths as columns

name: string Name to assign to collection

c: specdal.Collection object

4.3 Operators

Specdal's operators perform on both pandas and specdal objects. In the following operations, pandas series and dataframes correspond to specdal's spectrum and collection, respectively.

specdal.operators.**get_column_types**(*df*)
Returns a tuple (wvl_cols, meta_cols), given a dataframe.

Wavelength column is defined as columns with a numerical name (i.e. decimal). Everything else is considered metadata column.

specdal.operators.**stitch**(*series*, *method='max'*)
Stitch the regions with overlapping wavelength

series: pandas.Series object

method: string How to compute final value in case of overlap. "mean","median","min", or "max".

specdal.operators.**interpolate**(*series*, *spacing=1*, *method='slinear'*)
Interpolate the array into given spacing

series: pandas.Series object

spacing: int wavelength spacing to interpolate at (in nm)

method: string "slinear" or "cubic"

specdal.operators.**derivative**(*series*)
Calculate the spectral derivative. Not Implemented Yet.

specdal.operators.**proximal_join**(*base_df*, *rover_df*, *on='gps_time_tgt'*, *direction='nearest'*)
Perform proximal join and return a new dataframe.

base_df: pandas.DataFrame DataFrame of reference measurements

rover_df: pandas.DataFrame DataFrame of target measurements

proximal: pandas.DataFrame object proximally processed dataset (rover_df / base_df)

As a side-effect, the rover dataframe is sorted by the key Both base_df and rover_df must have the column specified by on. This column must be the same type in base and rover.

specdal.operators.**jump_correct**(*series*, *splices*, *reference*, *method='additive'*)
Correct for jumps in non-overlapping wavelengths

splices: list list of wavelength values where jumps occur

reference: int position of the reference band (0-based)

4.4 Readers

Specdal's readers parse a variety of input formats into the common specdal.containers.spectrum.Spectrum data type. Readers are used internally py Specturm and Collection when constructed with the filename argument, but can also be used individually.

specdal.readers.**read**(*filepath*, *read_data=True*, *read_metadata=True*, *verbose=False*)
Calls a reader function based on the extension of the passed filename. .asd: read_asd .sig: read_sig .sed: read_sed .pico: read_pico

specdal.readers.asd.**read_asd**(*filepath*, *read_data=True*, *read_metadata=True*, *verbose=False*)
Read asd file for data and metadata

2-tuple of (pd.DataFrame, OrderedDict) for data, metadata

specdal.readers.sig.**read_sig**(*filepath*, *read_data=True*, *read_metadata=True*, *verbose=False*)
Read asd file for data and metadata

2-tuple of (pd.DataFrame, OrderedDict) for data, metadata

specdal.readers.sed.**read_sed**(*filepath*, *read_data=True*, *read_metadata=True*, *verbose=False*)
Read sed file for data and metadata

2-tuple of (pd.DataFrame, OrderedDict) for data, metadata

specdal.readers.pico.**read_pico**(*filepath*, *read_data=True*, *read_metadata=True*, *verbose=False*)
Read pico file for data and metadata

2-tuple of (pd.DataFrame, OrderedDict) for data, metadata

4.5 Filters

Specdal's filters operate on Collection objects, splitting them into "good" and "bad" spectra based on certain criteria.

specdal.filters.**filter_std**(*collection*, *wavelength0*, *wavelength1*, *std_thresh*, *group='mean'*)
Filter the spectra from collection that have a standard deviation outside a certain threshold.

collection: specdal.containers.collection.Collection the collection to filter

wavelength0: float the starting wavelength to filter

wavelength1: float the ending wavelength to filter

std_thresh: float remove spectra outside of std_thresh standard deviations from the mean

group: string if there are multiple data points between wavelength0 and wavelength1, average them this way. Options: "mean", "median", "min", "max"

good: specdal.containers.Collection A new collection made of the spectra that passed the filter

bad: specdal.containers.Collection A new collection made of the spectra that failed the filter

specdal.filters.**filter_white**(*collection*, *wavelength0=0*, *wavelength1=10000*, *group='mean'*)
Filter white reference spectra from collection

good: specdal.containers.Collection A new collection made of the spectra that passed the filter

bad: specdal.containers.Collection A new collection made of the spectra that failed the filter

specdal.filters.**filter_threshold**(*collection*, *wavelength0*, *wavelength1*, *low*, *high*, *group='mean'*)
Filter the spectra from collection that have a value outside of (low,high). Parameters ———- collection: specdal.containers.collection.Collection

the collection to filter

wavelength0: float the starting wavelength to filter

wavelength1: float the ending wavelength to filter

low: float minimum allowed value between wavelength0 and wavelength1

high: float maximum allowed value between wavelength0 and wavelength1

group: string if there are multiple data points between wavelength0 and wavelength1, average them this way. Options: “mean”, “median”, “min”, “max”

good: specdal.containers.Collection A new collection made of the spectra that passed the filter

bad: specdal.containers.Collection A new collection made of the spectra that failed the filter

4.6 GUI

Specdal provides a Tkinter-based GUI for plotting and manually flagging spectra from Collections.

class specdal.gui.viewer.**ColorPickerDialog**(*parent*, *start_color=(0, 0, 0)*)

class specdal.gui.viewer.**PlotConfigDialog**(*parent*, *xlim=(0, 1)*, *ylim=(0, 1)*, *title=”*, *xlabel=”*, *ylabel=”*)

class specdal.gui.viewer.**ToolBar**(*canvas_*, *parent*, *ax*)

home()
Override home method to return to home of most recent plot

class specdal.gui.viewer.**Viewer**(*parent*, *collection=None*, *with_toolbar=True*)

jump_correct()
Only performs jump correction on 1000 and 1800 wvls and 1 reference

save_flag()
save flag to self.flag_filepath

save_flag_as()
modify self.flag_filepath and call save_flag()

stitch()
Can’t stitch one spectrum and plot the collection

update()
Update the plot

update_selected(*to_add=None*)
Update, only on flaged

CHAPTER 5

Specdal Pipeline Script

Specdal provides a command line script `specdal_pipeline` for batch processing of spectral data files in a directory. A typical input to `specdal_pipeline` is a directory containing spectral files (i.e. .asd files), which will be converted into .csv files and figures of spectra. User can provide arguments to customize the processing operations (i.e. jump correction, groupby) and output (i.e. .csv file of group means). This page describes the usage and provides examples.

5.1 Usage

```
usage: specdal_pipeline [-h] [--proximal_reference PATH] [-o PATH]
                        [-op PREFIX] [-of] [-od] [-oi] [-i {slinear,cubic}]
                        [-is SPC] [-s {mean,median,min,max}] [-j {additive}]
                        [-js WVL [WVL ...]] [-jr REF] [-g] [-gs S]
                        [-gi [I [I ...]]] [-gmean] [-gmedian] [-gstd]
                        [-fstd wl0 wl1 n_std [wl0 wl1 n_std ...]]
                        [-fthresh wl0 wl1 LO HI [wl0 wl1 LO HI ...]] [-fwhite]
                        [-fg method] [-fo set] [-yl ymin ymax] [-q] [-f]
                        INPUT_PATH
```

5.2 Command Line Arguments

positional arguments: `INPUT_PATH` directory containing input files

optional arguments: `-h, --help` show this help message and exit

`--proximal_reference PATH` directory containing proximal reference spectral files

`-o PATH, --output_dir PATH` directory to store the csv files and figures

`-op PREFIX, --prefix PREFIX` option to specify prefix for output dataset files

`-of, --omit_figures` option to omit output png figures

`-od, --omit_data` option to omit output csv files

`-oi, --omit_individual` option to omit output of individual csv file for each spectrum file

`-i {slinear,cubic}, --interpolate {slinear,cubic}` specify the interpolation method. method descriptions can be found on scipy docs: https://docs.scipy.org/doc/scipy-0.19.1/reference/generated/scipy.interpolate.interp1d.html

`-is SPC, --interpolate_spacing SPC` specify desired spacing for interpolation in nanometers

`-s {mean,median,min,max}, --stitch {mean,median,min,max}` specify overlap stitching method; not necessary if data at detector edges does not overlap

`-j {additive}, --jump_correct {additive}` specify jump correction method;

`-js WVL [WVL ...], --jump_correct_splices WVL [WVL ...]` wavelengths of jump locations

`-jr REF, --jump_correct_reference REF` specify the reference detector (e.g. VNIR is 1, SWIR1 is 2)

`-g, --group_by` create groups using filenames‘‘

`-gs S, --group_by_separator S` specify filename separator character to define groups

`-gi [I [I ...]], --group_by_indices [I [I ...]]` specify the indices of the split filenames to define a group

`-gmean, --group_mean` calculate group means and append to group figures

`-gmedian, --group_median` calculate group median and append to group figures

`-gstd, --group_std` calculate group standard deviation and append to group figures

`-fstd wl0 wl1 n_std [wl0 wl1 n_std ...], --filter_std wl0 wl1 n_std [wl0 wl1 n_std` Remove spectra from dataset with a pct_reflect over n_std away from the mean between wavelengths wl0 and wl1. Can specify multiple sets of wavenumber ranges and thresholds

`-fthresh wl0 wl1 LO HI [wl0 wl1 LO HI ...], --filter_threshold wl0 wl1 LO HI [wl0 w` Remove spectra from the dataset with a pct_reflect outside (LO,HI)in the wavenumber range wl0 wl1. Can specify multiple sets of wavenumber ranges and thresholds

`-fwhite, --filter_white` Remove white reference spectra from dataset

`-fg method, --filter_group method` How to combine the wavelengths selected by –filter_group.

`-fo set, --filter_on set` What subset of the data to apply filter on (collection, group or both)

`-yl ymin ymax, --ylim ymin ymax` Force the y axis of plots to display between ymin and ymax

`-q, --quiet`

`-f, --force` if output path exists, remove previous output and run

5.3 Examples

For a description of all command line arguments: `specdal_pipeline --help`.

To produce an individual plot and textfile for every spectrum file in directory `/path/to/spectra/` and store the results in `specdal_output/`: `specdal_pipeline -o specdal_output /path/to/spectra/`

To only output whole-dataset images and files: `specdal_pipeline -oi -o specdal_output /path/to/spectra/`

To only output images, with no data files: `specdal_pipeline -od -o specdal_output /path/to/spectra/`

To group input files by the first 3 underscore-separated components of their filename (such that `foo_bar_baz_001.asd` and `foo_bar_baz_002.asd` will appear in one group, and `foo_bar_qux_001.asd` in another): `specdal_pipeline -g -gi 0 1 2 -- /path/to/spectra/`

To also output the mean and median of every group of spectra: `specdal_pipeline -g -gi 0 1 2 -gmean -gmedian /path/to/spectra/`

To remove all white reference spectra from the output dataset (leaves input files intact): `specdal_pipeline --filter_white /path/to/spectra/`

5.4 Filtering (WIP)

specdal_pipeline also provides the option to automatically filter spectra out of the dataset. This feature is not fully tested and may cause issues.

To remove all spectra with a 750-1200 nm reflectance that is greater than 1 standard deviation from the mean, or with a 500-600 nm reflectance that is greater than 2 standard devations from the mean:

```
specdal_pipeline --filter_std 750 1200 1 500 600 2 -- /path/to/spectra/
```

To perform the filtering above, and then group the remaining spectra by filename:

```
specdal_pipeline --filter_std 750 1200 1 500 600 2 -g -gi 0 1 2 /path/to/spectra/
```

To group the spectra by filename, and then perform filtering on each group:

```
specdal_pipeline --filter_std 750 1200 1 500 600 2 -g -gi 0 1 2 --filter_on group /path/to/spectra/
```

CHAPTER 6

Specdal Info Script

6.1 Usage

```
usage: specdal_info [-h] [--raw] [--list_measure_types]
                    [--list_metadata_fields] [--measure_type MEASURE_TYPE]
                    [--metadata [FIELD [FIELD ...]]] [-n N] [-d]
                    FILE [FILE ...]
```

6.2 Command Line Arguments

positional arguments: `FILE` input directory containing input files

optional arguments: `-h, --help` show this help message and exit

`--raw` output raw dataframe and metadata and exit

`--list_measure_types` list measurement types and exit

`--list_metadata_fields` list metadata fields and exit

`--measure_type MEASURE_TYPE` type of measurement to read

`--metadata [FIELD [FIELD ...]]` specify metadata fields to display

`-n N, --N N` number of spectra to display from head and tail

`-d, --debug`

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

T

U

V