SpecDAL Documentation

Release 1.0

Prabu Ravindran, Clayton Kingdon

Mar 09, 2017

Contents

1 Guide **3**

1.1 Installation . . . 3
1.1.1 Requirements . . . 3
1.1.2 Recommended How To . . . 3
1.2 Classes . . . 4
1.2.1 SdalSpectrum . . . 4
1.2.2 SdalDataset . . . 6
1.2.3 SdalCollection . . . 7
1.2.4 SdalSession . . . 8
1.2.5 Readers . . . 8
1.2.5.1 AsdReader . . . 8
1.2.5.2 SedReader . . . 8
1.2.5.3 SigReader . . . 9
1.2.5.4 ReaderUtils . . . 10
1.2.6 Resamplers . . . 11
1.2.6.1 LinearResampler . . . 11
1.2.6.2 CubicSplineResampler . . . 11
1.2.6.3 ResamplerUtils . . . 12
1.2.7 Splitters . . . 13
1.2.7.1 NondecreasingWavelengthsSplitter . . . 13
1.2.7.2 AtSpecifiedWavelenthsSplitter . . . 13
1.2.7.3 IntervalOfWavelengthsSplitter . . . 14
1.2.7.4 SplitterUtils . . . 15
1.2.8 Stitchers . . . 16
1.2.8.1 OverlapAveragingStitcher . . . 16
1.2.8.2 OverlapBlendingStitcher . . . 16
1.2.8.3 JumpCorrectingStitcher . . . 17
1.2.8.4 ClosestApproachStitcher . . . 18
1.2.9 Column operations . . . 18
1.2.9.1 ColumnExponential . . . 18
1.2.9.2 ColumnLogarithm . . . 19
1.2.9.3 ColumnOffset . . . 20
1.2.9.4 ColumnPolynomial . . . 20
1.2.9.5 ColumnPower . . . 21
1.2.9.6 ColumnScale . . . 22
1.2.9.7 ColumnUtils . . . 22

1.2.10 Helpers . . . 23
1.2.10.1 PathParts . . . 23
1.2.10.2 SdalUtils . . . 23
1.2.10.3 SpectrumUtils . . . 24
1.3 Scripts . . . 25
1.4 Extras . . . 25

Python Module Index **27**

SpecDAL is Python package for loading and manipulating field spectroscopy data. It currently supports importing data produced by spectrometers from ASD, SVC and SED. It can also do standard processing of the data such as: resampling, jump correction, overlap handling, wavelength masking and wrangling the data associated with a project/campaign into csv files.

CHAPTER 1

Guide

Installation

Requirements

Python3: SpecDAL requires Python3 along with numpy, pandas and matplotlib. Anaconda provides a installer for Python3 that makes life easier. Anaconda can be downloaded here: <https://www.continuum.io/>. Please use the 64-bit version for your operating system.

Git: Git is required if you want to clone the SpecDAL repos on your computer. This is the recommended way to get SpecDAL. Git can be downloaded here: <https://git-scm.com/>.

Recommended How To

1. Install Python 3 and other required packages using Ananconda.
2. Install git so that we can clone SpecDAL.
3. Getting the **specdal** package.
   (a) Create a directory called SpecDAL.
   (b) Change into this directory (e.g. cd SpecDAL).
   (c) At the command line type:

```
git clone https://github.com/SpecDAL/specdal.git
```

Classes

SdalSpectrum

class sdal_spectrum.**SdalSpectrum**

Class to hold data and metadata for a spectrum.

Numerical data is stored in a dictionary with a provided label used as key. The associated value is a list of 2-column numpy.ndarrays.

Metadata is stored in a dictionary with a provided label used as key.

While there is no requirement that the keys for data and metadata match for an object, it is recommended that they match.

add_data(*label*, *waves=<Mock name='mock.array()' id='140464197230480'>*, *refls=<Mock name='mock.array()' id='140464197230480'>*, *twocol=<Mock name='mock.array()' id='140464197230480'>*)

Add numerical data to the object.

Parameters

- **label** (*string*) – The label for provided numerical data.
- **waves** (*array_like*) – Wavelengths.
- **refls** (*array_like*) – Reflectances.
- **twocol** (*array_like*) – Data matrix with wavelengths in column one and reflectances in column two.

Note:

1. If provided *waves* and *refls* must be same length.
2. If more than 2 columns in twocol, first 2 columns used.

---

add_metadata(*key_val*)

Add metadata to object.

Parameters

- **label** (*string*) – Label for provided metadata.
- **key_val** (*tuple*) – Metadata key-value pairs as a tuple.

del_data(*label*)

Delete specified label and associated data.

Parameters **label** (*string*) – Data label to be removed.

get_data(*label*)

Get numerical data from object.

Parameters **label** (*string*) – Label for requested numerical data.

Returns A list of two column numpy.ndarray. Column[0]: wavelengths, Column[1]: reflectances.

Return type list

Note: A list of two column `numpy.ndarray` is returned as there can be multiple pieces associated with a label. For example the pieces could be ones with increasing wavelengths when overlap is being handled.

get_data_labels()
Get data labels for the object. Order in which labels were added is maintained.

Returns A list of data labels (strings).

Return type list

get_metadata(*label*)
Get metadata from object.

Parameters **label** (*string*) – Label for requested metadata.

Returns Value for key.

Return type value

get_metadata_labels()
Get metadata labels for the object.

Order in which labels were added is maintained.

Returns A list of metadata labels (strings).

Return type list

get_name()
Get name for spectrum object.

Returns The name of the spectrum object.

Return type string

is_valid_data_label(*label*)
Check if specified label is a valid data label.

Parameters **label** (*string*) – Label to check.

Returns True if valid, False otherwise.

Return type bool

is_valid_meta_label(*label*)
Check if specified label is a valid meta label.

Parameters **label** (*string*) – Label to check.

Returns True if valid, False otherwise.

Return type bool

print_data(*label=''*, *num_head=5*, *num_tail=5*)
Print numerical data of object.

Parameters

- **label** (*string*) – Label for requested numerical data. If empty, data associated with all labels are printed. Default="".
- **num_head** (*int*) – Number of values at head to print. Default=5.
- **num_tail** (*int*) – Number of values at tail to print. Default=5.

Note: Mainly for debugging/testing purposes.

print_metadata(*label=''*)
Print metadata for object.

Parameters **label** (*string*) – Label for requested metadata. If empty, metadata for all labels are printed. Default="".

Note: Mainly for debuggin/testing purposes.

set_name(*name*)
Set name for spectrum object.

Parameters **name** (*string*) – The name for the spectrum object.

Note:

1. It is recommended to set name first before adding any data. This will lead to more useful warning/error messages.
2. This is typically the filename. Spectrum from multi-spectrum file can be named by adding prefix/suffix to the filename.

SdalDataset

class sdal_dataset.**SdalDataset**
Class that will contain the dataset being processed. It contains *SdalCollection* objects which in turn contain *SdalSpectrum* objects.

add_spectrum(*spectrum*, *coll_name=''*)
Add spectrum to dataset.

Parameters

- **spectrum** (SdalSpectrum) – A SdalSpectrum object to be added
- **coll_name** (*string*) – Name of collection to which to add. If provided name is not already present a new SdalCollection is initialized and added. If *coll_name* is empty then the data labels in *spectrum* are used to see if there is already a collection with those data labels. If yes *spectrum* is added to that collection. If it is a new set of data labels a new collection with the generic name collection# is initialized and the spectrum added to it.

get_collection(*coll_name*)
Get collection object with the provided *coll_name*.

Parameters **coll_name** (*string*) – The collection name.

Returns 2-element tuple (collection, msg) collection is a SdalCollection object. msg is a list of error message strings. If something is wrong the returned tuple is (None, msg) where msg is non-empty list. All is well if msg is empty list.

Return type tuple

get_collection_names()
Get list of collection names.

Returns List of collection names.

Return type list

is_valid_collection(*coll_name*)
Check validity of collection name.

Parameters **coll_name** (*string*) – Name of collection to check

Returns Whether *coll_name* is valid.

Return type bool

print_summary(*coll_name=''*)
Prints details of the dataset.

Parameters **coll_name** (*string*) – The name of collection to print. If empty all collections are printed.

update_mask(*coll_name*, *mask_vals={}*)
Update mask in a collection.

Parameters

- **coll_name** (*string*) – The collection name
- **mask_vals** (*dict*) – Keys are names of spectrum and values are True/False booleans.

Returns List of error message strings. All is well if empty.

Return type list

SdalCollection

class sdal_collection.**SdalCollection**
Class to hold a collection of spectra along with a mask.

add_spectrum(*spectrum*, *mask_val=True*)
Add a spectrum and its mask.

Parameters

- **spectrum** (SdalSpectrum) – A SdalSpectrum object. *It must have name.* If no name for spectrum it is silently skipped.
- **mask_val** (*boolean*) – The mask value for the spectrum.

fill_mask(*mask_val=True*)
Fill mask values.

Parameters **mask_val** (*bool*) – The value to set. Default=True.

get_mask()
Get mask values from an OrderedDict. The order of mask values is the same as spectrum order.

Returns List of mask values (booleans)

Return type list

get_spectrum(*name*)
Get a particular by name.

Parameters **name** (*string*) – A spectrum name.

Returns A tuple (spectrum, msg). *spectrum* is a SdalSpectrum object. *msg* is a list of error message strings. Is empty all is well. If somethins is invalid a None is returned for *spectrum*

Return type tuple

get_spectrums()
Get spectrums list. The order is same as that of self.get_mask.

Returns List of SdalSpectrum objects.

Return type list

print_summary(*header=''*)
Prints a summary

Parameters **header** (*string*) – A message to be printed.

update_mask(*mask_vals={}*)
Update mask values for specific spectrums.

Parameters **mask_vals** (*dict*) – A dictionary with spectrum_name–>boolean. Silently skips invalid keys.

SdalSession

Readers

AsdReader

class asd_reader.**AsdReader**
Class to read in asd files.

load_spectrum(*filepath*)
Loads contents of .asd file into a SdalSpectrum object.

Parameters **filepath** (*string*) – Full path to .asd file.

Returns

2-element tuple: (*spectrum*, *msg*).

1. **'spectrum' is 'SdalSpectrum' object with data from file.** None is returned if something goes wrong.
2. 'msg' is a list of error message strings. If empty all is well.

Return type tuple

run(*dataset*, *op_key*, *args_dict*)
Loads SdalSpectrum objects into a SdalDataset.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this reader (–read.asd).
- **args_dict** (*dict*) – The arguments that were passed for the read operation.

SedReader

class sed_reader.**SedReader**
Class to read in sed files.

load_spectrum(*filepath*)
Loads contents of .sed/.raw file into a SdalSpectrum object.

Parameters **filepath** (*string*) – Full path to .sed/.raw file.

Returns

2-element tuple: (*spectrum*, *msg*).

1. **'spectrum' is 'SdalSpectrum' object with data from file.** None is returned if something goes wrong.
2. 'msg' is a list of error message strings. If empty all is well.

Return type tuple

Note:

1.The mapping to data matrix labels is in sed_columns_mapping.txt.

---

run(*dataset*, *op_key*, *args_dict*)
Loads SdalSpectrum objects into a SdalDataset.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this reader (–read.asd).
- **args_dict** (*dict*) – The arguments that were passed for the read operation.

SigReader

class sig_reader.**SigReader**
Class to read in sig files.

load_spectrum(*filepath*)
Loads contents of .sig file into a SdalSpectrum object.

Parameters **filepath** (*string*) – Full path to .sig file

Returns

2-element tuple: (*spectrum*, *msg*).

1. **'spectrum' is 'SdalSpectrum' object with data from file.** None is returned if something goes wrong.
2. 'msg' is a list of error message strings. If empty all is well.

Return type tuple

Note:

1.The mapping to data matrix labels is in sig_columns_mapping.txt.

---

run(*dataset*, *op_key*, *args_dict*)
Loads SdalSpectrum objects into a SdalDataset.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.

- **op_key** (*string*) – The key used to select this reader (–read.asd).
- **args_dict** (*dict*) – The arguments that were passed for the read operation.

ReaderUtils

class reader_utils.**ReaderUtils**
Utilities class for readers.

dedup_wavelengths(*nptwocol*)
Remove duplicate/repeated wavelengths.

It assumes that repeated values are consecutive. i.e. x, x, y -> x, y and x, y, x -> x, y, x.

Parameters **nptwocol** (*numpy.ndarray*) – A numpy array with wavelengths in first column.

Returns A two column matrix with with consecutively appearing duplicated wavelengths removed.

Return type numpy.ndarray

get_filepaths(*extension*, *input_dirs=[]*, *input_files=[]*)
Get list of files with specified extension from provided lists of directories and files.

Parameters

- **extension** (*string*) – The required file extension.
- **input_dirs** (*list*) – List of directory name strings.
- **input_files** (*list*) – List of file name strings.

Returns

2-element tuple: (*file_paths*, *msg*).

1. *file_paths* is a list of strings
2. 'msg' is a list of error message strings. If empty all is well.

Return type tuple

is_valid_filepath(*filepath*, *extensions=[]*)
Check validity of *filepath*.

Parameters

- **filpath** (*string*) – Path to file.
- **extensions** (*list*) – List of valid filename extensions. If empty just validity of path is checked.

Returns A list of error message strigs. All is well if empty.

Return type list

Note: Existence and correct extension (if *extensions* non empty) checked.

Resamplers

LinearResampler

class linear_resampler.**LinearResampler**
Bases: object

Resample using linear interpolation between wavelengths.

resample(*spectrum*, *src_label*, *dst_label*, *req_waves=<Mock name='mock.array()' id='140464197230480'>*, *del_src=False*)
Resample the spectrum data using linear interpolation.

Parameters

- **spectrum** (SdalSpectrum) – SdalSpectrum object for input and output.
- **src_label** (*string*) – Label for input data.
- **dst_label** (*string*) – Label for output data.
- **req_waves** (*numpy.ndarray*) – Wavelengths at which to resample (1D array). If this is empty resampling happens at 1nm wavelengths within range of input.
- **del_src** (*bool*) – If True *src_label* is deleted. Default is False.

Returns List of error messages. If empty all is well.

Return type list

Note:

1. If *req_waves* are provided they are clamped to range of input spectrum.
2. For each piece of *src_label* resampling happens at *req_waves* within the range of the piece and added to *dst_label*.
3. Wavelengths in each *src_label* piece must be increasing.

---

run(*dataset*, *op_key*, *args_dict*)
Performs linear resampling of specified spectrums.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this resampler (–resample.linear).
- **args_dict** (*dict*) – The arguments that were passed for resample operation.

Returns List of error messages. If empty all is well.

Return type list

CubicSplineResampler

class cubic_spline_resampler.**CubicSplineResampler**
Bases: object

Resample using cubic hermite spline interpolation between wavelengths.

resample(*spectrum*, *src_label*, *dst_label*, *req_waves=<Mock name='mock.array()' id='140464197230480'>*, *del_src=False*)
Resample the spectrum data using linear interpolation.

Parameters

- **spectrum** (SdalSpectrum) – SdalSpectrum object for input and output.
- **src_label** (*string*) – Label for input data.
- **dst_label** (*string*) – Label for output data.
- **req_waves** (*numpy.ndarray*) – Wavelengths at which to resample (1D array). If this is empty resampling happens at 1nm wavelengths within range of input.
- **del_src** (*bool*) – If True *src_label* is deleted. Default is False.

Returns List of error messages. If empty all is well.

Return type list

Note:

1. If *req_waves* are provided they are clamped to range of input spectrum.
2. For each piece of *src_label* resampling happens at *req_waves* within the range of the piece and added to *dst_label*.
3. Wavelengths in each *src_label* piece must be increasing.

---

run(*dataset*, *op_key*, *args_dict*)
Performs cubic spline resampling of specified spectrums.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this resampler (–resample.cubic_spline).
- **args_dict** (*dict*) – The arguments that were passed for resample operation.

Returns List of error messages. If empty all is well.

Return type list

ResamplerUtils

class resampler_utils.**ResamplerUtils**
Utilities for resampling classes.

get_taps(*dm*, *req_waves=<Mock name='mock.array()' id='140464197230480'>*)
Valid wavelengths/taps that are clamped to *dm* range.

Parameters

- **dm** (*numpy.ndarray*) – 2-column data matric with wavelengths in first column.
- **req_waves** (*numpy.ndarray*) – List of wavelengths. Default is empty. If empty, one nm wavelengths in range of *dm* returned. If non-empty, wavelengths are clamped to range of *dm*.

Returns Valid wavelengths/taps that are clamped to *dm* range.

Return type numpy.ndarray

Splitters

NondecreasingWavelengthsSplitter

class nondecreasing_wavelengths_splitter.**NondecreasingWavelengthsSplitter**
Split into pieces with non-decreasing wavelengths.

run(*dataset*, *op_key*, *args_dict*)
Splits spectrum data matrices into nondecreasing wavelength pieces.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this splitter (–split.nondecreasing_wavelengths).
- **args_dict** (*dict*) – The arguments that were passed for resample operation.

Returns List of error messages. If empty all is well.

Return type list

split(*spectrum*, *src_label*, *dst_label*, *del_src=False*)

Parameters

- **spectrum** (SdalSpectrum) – SdalSpectrum object which will be modified.
- **src_label** (*string*) – Label of input data to splitting operation. This is label used to access relevant data in *s*.
- **dst_label** (*string*) – Label of split pieces when inserted into *s*.
- **del_src** (*bool*) – If True src_label is deleted. Default is False.

Note:

1. Wavelength spacing can be arbitrary.
2. If multiple data arrays are associated with *src_label*, each array is split into pieces of increasing wavelengths.

AtSpecifiedWavelenthsSplitter

class at_specified_wavelengths_splitter.**AtSpecifiedWavelengthsSplitter**
Split into pieces at specified wavelengths.

run(*dataset*, *op_key*, *args_dict*)
Splits spectrum data matrices at specified wavelengths.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this splitter (–split.at_specified_wavelengths).
- **args_dict** (*dict*) – The arguments that were passed for resample operation.

Returns List of error messages. If empty all is well.

Return type list

split (*spectrum*, *src_label*, *dst_label*, *split_waves=[]*, *split_label=''*, *del_src=False*)

Parameters

- **spectrum** (SdalSpectrum) – SdalSpectrum object which will be modified.
- **src_label** (*string*) – Label of input data to splitting operation. This is label used to access relevant data in *s*.
- **dst_label** (*string*) – Label of split pieces when inserted into *s*.
- **split_waves** (*list*) – List of wavelengths at which to break the input spectrum. The wavelengths will be forced to be increasing.

  If the input wavelengths are $[s, e]$ and split_waves are $[a, b]$ then the pieces would be the ranges $[s, a']$, $[a'', b']$, $[b'', e]$.

  a', a'', b', b'' are closest wavelengths to a and b such that $a' \leq a$, $b' \leq b$, $a'' > a$ and $b'' > b$.

  If a specified wavelength is outside the range of input wavelengths no split happens at that wavelength.
- **split_label** (*string*) – Metadata label to access from *spectrum* to get split_waves.
- **del_src** (*bool*) – If True src_label is deleted. Default is False.

Returns List of error message strings. All is well if empty.

Return type list

Note:

1. Wavelengths must be increasing.
2. If multiple data arrays are associated with *src_label*, each array is split at specified wavelengths.
3. One of *split_waves* or *split_label* must be specified.
4. Entries in split_waves are uniquified before splitting spectrum.

IntervalOfWavelengthsSplitter

class interval_of_wavelengths_splitter.**IntervalOfWavelengthsSplitter**

Split into pieces based on wavelength intervals.

run (*dataset*, *op_key*, *args_dict*)

Splits spectrum data matrices with wavelength intervals.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this splitter (–split.interval_of_wavelengths).
- **args_dict** (*dict*) – The arguments that were passed for resample operation.

Returns List of error messages. If empty all is well.

Return type list

split (*spectrum*, *src_label*, *dst_label*, *req_ranges*, *del_src=False*)

Parameters

- **spectrum** (SdalSpectrum) – SdalSpectrum object which will be modified.
- **src_label** (*string*) – Label of input data to splitting operation. This is label used to access relevant data in s.
- **dst_label** (*string*) – Label of split pieces when inserted into s.
- **req_ranges** (*list*) – List of wavelength ranges (2-tuples) to extract from input.

  If the tuple (a, b) is a specified wavelength range:

  $s < a < b < e$, range extracted is $[a', b']$.

  $a < s < b < e$, range extracted is $[s, b']$.

  $s < a < e < b$, range extracted is $[a', e]$.

  $a < b < s < e$, nothing is extracted.

  $s < e < a < b$, nothing is extracted.

  a' and b' are closest wavelengths to a and b respectively such that $a' >= a$ and $b' <= b$.
- **del_src** (*bool*) – If True src_label is deleted. Default is False.

Returns List of error message strings. All is well if empty.

Return type list

Note:

1. Wavelengths must be increasing.
2. If multiple data arrays are associated with *src_label*, pieces within specified ranges are extracted from each one.

SplitterUtils

class splitter_utils.**SplitterUtils**

Utilities for splitting classes.

extract_pieces(*dm*, *ends*)

Extract sub-matrices from *dm* using *ends*. If *ends* = [a, b, c, d] then the pieces would be [dm[a:b, :], dm[b:c, :], dm[c:d, :]]. It is assumed that the elements in *ends* are all valid. 'ends[i-1]' and 'ends[i]' are the indices of the slice that we want. Only non-empty pieces are returned.

Parameters

- **dm** (*numpy.ndarray*) – A numpy matrix. This would be the 2-column data matrix.
- **ends** (*list*) – A list of indices which contain range of rows in *dm*. Entries in *ends* are employed in numpy slices.

Returns A list of 2-column numpy arrays.

Return type list

Stitchers

OverlapAveragingStitcher

class overlap_averaging_stitcher.**OverlapAveragingStitcher**
Stich by averaging values at overlapping wavelengths.

run(*dataset*, *op_key*, *args_dict*)
Performs overlap averaging stitching of specified spectrums.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this resampler (–stitch.overlap_averaging).
- **args_dict** (*dict*) – The arguments that were passed for stitch operation.

Returns List of error messages. If empty all is well.

Return type list

stitch(*spectrum*, *src_label*, *dst_label*, *del_src=False*)
Stitch the pieces by averaging values at overlapping wavelengths.

Parameters

- **spectrum** (SdalSpectrum) – Spectrum object.
- **src_label** (*string*) – Label for input data. The associated data must be 1nm spaced.
- **dst_label** (*string*) – Label for output data. Added to *spectrum*.
- **del_src** (*bool*) – If True src_label is deleted. Default is False.

Returns List of error messages. If empty all is well.

Return type list

Notes

1. Wavelengths need to be increasing and 1nm.
2. Piece wavelengths must be in non-decreasing order.

OverlapBlendingStitcher

class overlap_blending_stitcher.**OverlapBlendingStitcher**
Stitch by linearly blending values at overlapping wavelengths.

run(*dataset*, *op_key*, *args_dict*)
Performs overlap blending stitching of specified spectrums.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this resampler (–stitch.overlap_blending).
- **args_dict** (*dict*) – The arguments that were passed for stitch operation.

Returns List of error messages. If empty all is well.

Return type list

stitch (*spectrum*, *src_label*, *dst_label*, *del_src=False*)
Stitch by linearly blending values at overlapping wavelengths.

Parameters

- **spectrum** (SdalSpectrum) – Spectrum object.
- **src_label** (*string*) – Label for input data. The associated data must be 1nm spaced.
- **dst_label** (*string*) – Label for output data. Added to *spectrum*.
- **del_src** (*bool*) – If True src_label is deleted. Default is False.

Returns List of error messages. If empty all is well.

Return type list

Note:

1.Wavelengths spacing must be increasing and unit spaced.

2.Piece firstwavelengths must be in non-decreasing order.

JumpCorrectingStitcher

class jump_correcting_stitcher.**JumpCorrectingStitcher**
Stitch by jump correcting at specified wavelengths.

run (*dataset*, *op_key*, *args_dict*)
Performs jump correcting stitching of specified spectrums.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this resampler (–stitch.jump_correcting).
- **args_dict** (*dict*) – The arguments that were passed for stitch operation.

Returns List of error messages. If empty all is well.

Return type list

stitch (*spectrum*, *src_label*, *dst_label*, *stable_index=1*, *del_src=False*)
Stich by jump correcting at ends of pieces.

Parameters

- **spectrum** (SdalSpectrum) – Spectrum object.
- **src_label** (*string*) – Label for input data. The associated data must be 1nm spaced.
- **dst_label** (*string*) – Label for output data. Added to *spectrum*.
- **stable_index** (*int*) – The piece whose value is unchanged..
- **del_src** (*bool*) – If True src_label is deleted. Default is False.

Returns List of error messages. If empty all is well.

Return type list

Note:

1.Wavelengths spacing must be increasing and unit spaced.

2.Pieces must be "adjacent", so that after stitching they are 1 nm and increasing.

ClosestApproachStitcher

class closest_approach_stitcher.**ClosestApproachStitcher**
Stitch at wavelengths where reflectances of overlapping pieces are the closest.

run(*dataset*, *op_key*, *args_dict*)
Performs closest approach stitching of specified spectrums.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this resampler (–stitch.closest_approach).
- **args_dict** (*dict*) – The arguments that were passed for stitch operation.

Returns List of error messages. If empty all is well.

Return type list

stitch(*spectrum*, *src_label*, *dst_label*, *del_src=False*)
Stitch at wavelengths where reflectances of overlapping pieces are the closest.

Parameters

- **spectrum** (SdalSpectrum) – Spectrum object.
- **src_label** (*string*) – Label for input data. The associated data must be 1nm spaced.
- **dst_label** (*string*) – Label for output data. Added to *spectrum*.
- **del_src** (*bool*) – If True src_label is deleted. Default is False.

Returns List of error messages. If empty all is well.

Return type list

Note:

1.Wavelengths spacing must be increasing and unit spaced.

2.Piece first wavelengths must be in non-decreasing order.

Column operations

ColumnExponential

class column_exponential.**ColumnExponential**
Class wraps numpy to apply the exp function to a column.

exponential(*spectrum*, *src_label*, *src_col_index*, *dst_label*, *del_src=False*)
Computes exp of column elements.

Parameters

- **spectrum** (SdalSpectrum) – A *SdalSpectrum* object
- **src_label** (*string*) – The source data label.
- **src_col_index** (*int*) – Column index of source data matrix. Must be 0/1.
- **dst_label** (*string*) – The destination/result data label. Note that the result is added to *spectrum* (the source object).
- **del_src** (*bool*) – Whether to delete the source after computation. Default=False.

Returns List of error message strings. All is well if empty.

Return type list

run (*dataset*, *op_key*, *args_dict*)
Applies exp function to column elements of specified spectrums.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this op (–column_op.exponential).
- **args_dict** (*dict*) – The arguments that were passed for exponential operation.

Returns List of error messages. If empty all is well.

Return type list

ColumnLogarithm

class column_logarithm.**ColumnLogarithm**
Class wraps numpy to apply the log function to a column.

logarithm (*spectrum*, *src_label*, *src_col_index*, *dst_label*, *del_src=False*)
Computes log of column elements.

Parameters

- **spectrum** (SdalSpectrum) – A *SdalSpectrum* object
- **src_label** (*string*) – The source data label.
- **src_col_index** (*int*) – Column index of source data matrix. Must be 0/1.
- **dst_label** (*string*) – The destination/result data label. Note that the result is added to *spectrum* (the source object).
- **del_src** (*bool*) – Whether to delete the source after computation. Default=False.

Returns List of error message strings. All is well if empty.

Return type list

run (*dataset*, *op_key*, *args_dict*)
Applies log function to column elements of specified spectrums.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this op (–column_op.logarithm).
- **args_dict** (*dict*) – The arguments that were passed for logarithm operation.

Returns List of error messages. If empty all is well.

Return type list

ColumnOffset

class column_offset.**ColumnOffset**
Class that adds a scalar to every element of a column.

offset(*spectrum*, *offset*, *src_label*, *src_col_index*, *dst_label*, *del_src=False*)
Adds offset to column elements.

Parameters

- **spectrum** (SdalSpectrum) – A *SdalSpectrum* object
- **offset** (*float*) – The offset to apply.
- **src_label** (*string*) – The source data label.
- **src_col_index** (*int*) – Column index of source data matrix. Must be 0/1.
- **dst_label** (*string*) – The destination/result data label. Note that the result is added to *spectrum* (the source object).
- **del_src** (*bool*) – Whether to delete the source after computation. Default=False.

Returns List of error message strings. All is well if empty.

Return type list

run(*dataset*, *op_key*, *args_dict*)
Adds offset to column elements of specified spectrums.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this op (–column_op.offset).
- **args_dict** (*dict*) – The arguments that were passed for offset operation.

Returns List of error messages. If empty all is well.

Return type list

ColumnPolynomial

class column_polynomial.**ColumnPolynomial**
Class wraps numpy to apply polynomial mapping to column.

polynomial(*spectrum*, *coefficients*, *src_label*, *src_col_index*, *dst_label*, *del_src=False*)
Computes polynomial of column elements.

Parameters

- **spectrum** (SdalSpectrum) – A *SdalSpectrum* object
- **coefficients** (*list*) – List of coefficients of polynomial (floats).
- **src_label** (*string*) – The source data label.
- **src_col_index** (*int*) – Column index of source data matrix. Must be 0/1.

- **dst_label** (*string*) – The destination/result data label. Note that the result is added to *spectrum* (the source object).
- **del_src** (*bool*) – Whether to delete the source after computation. Default=False.

Returns List of error message strings. All is well if empty.

Return type list

run(*dataset*, *op_key*, *args_dict*)
Applies polynomial mapping to column elements of specified spectrums.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this op (–column_op.polynomial).
- **args_dict** (*dict*) – The arguments that were passed for polynomial operation.

Returns List of error messages. If empty all is well.

Return type list

ColumnPower

class column_power.**ColumnPower**
Class wraps numpy to apply pow function to column elements.

power(*spectrum*, *exponent*, *src_label*, *src_col_index*, *dst_label*, *del_src=False*)
Computes power of column elements.

Parameters

- **spectrum** (SdalSpectrum) – A *SdalSpectrum* object
- **exponent** (*float*) – The power.
- **src_label** (*string*) – The source data label.
- **src_col_index** (*int*) – Column index of source data matrix. Must be 0/1.
- **dst_label** (*string*) – The destination/result data label. Note that the result is added to *spectrum* (the source object).
- **del_src** (*bool*) – Whether to delete the source after computation. Default=False.

Returns List of error message strings. All is well if empty.

Return type list

run(*dataset*, *op_key*, *args_dict*)
Computes powers of column elements of specified spectrums.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this op (–column_op.power).
- **args_dict** (*dict*) – The arguments that were passed for power operation.

Returns List of error messages. If empty all is well.

Return type list

ColumnScale

class column_scale.**ColumnScale**
Class multiplees elements of column by scalar.

run(*dataset*, *op_key*, *args_dict*)
Multiples column elements of specified spectrums with scalar.

Parameters

- **dataset** (SdalDataset) – A SdalDataset object.
- **op_key** (*string*) – The key used to select this op (–column_op.scale).
- **args_dict** (*dict*) – The arguments that were passed for scale operation.

Returns List of error messages. If empty all is well.

Return type list

scale(*spectrum*, *scale*, *src_label*, *src_col_index*, *dst_label*, *del_src=False*)
Computes power of column elements.

Parameters

- **spectrum** (SdalSpectrum) – A *SdalSpectrum* object
- **scale** (*float*) – The multiplication factor.
- **src_label** (*string*) – The source data label.
- **src_col_index** (*int*) – Column index of source data matrix. Must be 0/1.
- **dst_label** (*string*) – The destination/result data label. Note that the result is added to *spectrum* (the source object).
- **del_src** (*bool*) – Whether to delete the source after computation. Default=False.

Returns List of error message strings. All is well if empty.

Return type list

ColumnUtils

class column_utils.**ColumnUtils**
Class of utilities for column ops.

is_valid(*spectrum*, *col_index*, *src_label*, *dst_label*)
Perform set of checks related to column ops.

Parameters

- **spectrum** (SdalSpectrum) – A *SdalSpectrum* object.
- **col_index** (*int*) – Column index of data matrices. Must be 0/1.
- **src_label** (*string*) – Source data label.
- **dst_label** (*string*) – Destination/Result data label.

Returns A list of error message strings. All is well if empty.

Return type list

Helpers

PathParts

class path_parts.**PathParts**(*apath*)

A class that splits a file path into components. Based on code in the Python Cookbook.

directory()

Get directory portion.

Returns Prefix of filename in path.

Return type string

extension()

Get extension.

Returns Extension in filename. The . is not part of what is returned.

Return type string

filename()

Get filename portion.

Returns File name portion of path.

Return type string

parts()

Returns the pieces into which the path was split.

Returns List of all pieces of path.

Return type list

SdalUtils

class sdal_utils.**SdalUtils**

A class for common utilies for SpecDAL.

check_args_dict(*args_dict={}*, *required=[]*, *required_one_of=[]*, *optional=[]*, *class_name=''*)

Checks if args_dict has valid entries.

Parameters

- **args_dict** (*dict*) – Dictionary of specified args, of the form key->[list] Even single values are in list of length 1.
- **required** (*list*) – List of tuples.
  1. If tuple is (t_0,), t_0 must be in args_dict.
  2. If tuple is (t_0, t_1) must be present in args_dict and it must be given one of values in t_1.
- **required_one_of** (*list*) – List of tuples.

  The tuples can be any sized. If the tuple is (t_0, t_1, ..., t_n) then atleast one of the t_i must be in args_dict.
- **optional** (*list*) – List of optional arguments.

Returns A list of error messages. If empty all is well.

Return type list

Note:

1. *optional* is used to contruct the list of all possible arguments. So if an unknown argument is specified it is caught.

SpectrumUtils

class spectrum_utils.**SpectrumUtils**(*spectrum*, *label=''*, *meta_label=''*, *piece_index=0*)

Class for validations/checks on SdalSpectrum objects.

are_wavelengths_increasing()

Queries if the wavelengths are increasing. Assumes that the data label is valid. If wavelengths are increasing, empty list returned.

Returns List of messages if check fails. Empty list otherwise.

Return type list

are_wavelengths_unit_spaced()

Queries if the wavelengths are unit spaced. Assumes that the data label is valid. If wavelengths are unit spaced, empty list returned.

Returns List of messages if check fails. Empty list otherwise.

Return type list

has_one_data_array()

Queries if exactly one array associated with data_label. Assumes that the data label is valid. If exactly one data array for label, empty list returned.

Returns List of messages if check fails. Empty list otherwise.

Return type list

is_label_valid()

Queries if the data label is valid. If valid label, empty list returned.

Returns List of messages if check fails. Empty list otherwise.

Return type list

is_meta_label_valid()

Queries if the meta label is valid. If valid label, empty list returned.

Returns List of messages if check fails. Empty list otherwise.

Return type list

is_piece_index_valid()

Queries if piece_index is valid. Assumes that the data label is valid. If valid empty list returned.

Returns List of messages if check fails. Empty list otherwise.

Return type list

is_piece_order_valid()

Queries if piece ordering is valid. Assumes data label is valid. Assumes increasing wavelengths. Checks of dm[i][0, 0] <= dm[i + 1][0, 0] <= .... If valid empty list returned.

Returns List of messages if check fails. Empty list otherwise.

Return type list

Scripts

SpecDAL comes along with a script that implements an example pipeline. This pipeline script reads a desired collection of spectrum files and produces spectra with one nanometer spaced wavelengths. Various arguments can be used to alter the behavior of this script. This script can be run from the commandline.

A developer can use the above script as an example and employ the classes in SpecDAL to implement their own processing pipelines.

Extras

The gui maybe goes here.

Python Module Index

a

asd_reader, 8
at_specified_wavelengths_splitter, 13

c

closest_approach_stitcher, 18
column_exponential, 18
column_logarithm, 19
column_offset, 20
column_polynomial, 20
column_power, 21
column_scale, 22
column_utils, 22
cubic_spline_resampler, 11

i

interval_of_wavelengths_splitter, 14

j

jump_correcting_stitcher, 17

l

linear_resampler, 11

n

nondecreasing_wavelengths_splitter, 13

o

overlap_averaging_stitcher, 16
overlap_blending_stitcher, 16

p

path_parts, 23

r

reader_utils, 10
resampler_utils, 12

s

sdal_collection, 7
sdal_dataset, 6
sdal_spectrum, 4
sdal_utils, 23
sed_reader, 8
sig_reader, 9
spectrum_utils, 24
splitter_utils, 15

Index

A

add_data() (sdal_spectrum.SdalSpectrum method), 4
add_metadata() (sdal_spectrum.SdalSpectrum method), 4
add_spectrum() (sdal_collection.SdalCollection method), 7
add_spectrum() (sdal_dataset.SdalDataset method), 6
are_wavelengths_increasing() (spectrum_utils.SpectrumUtils method), 24
are_wavelengths_unit_spaced() (spectrum_utils.SpectrumUtils method), 24
asd_reader (module), 8
AsdReader (class in asd_reader), 8
at_specified_wavelengths_splitter (module), 13
AtSpecifiedWavelengthsSplitter (class in at_specified_wavelengths_splitter), 13

C

check_args_dict() (sdal_utils.SdalUtils method), 23
closest_approach_stitcher (module), 18
ClosestApproachStitcher (class in closest_approach_stitcher), 18
column_exponential (module), 18
column_logarithm (module), 19
column_offset (module), 20
column_polynomial (module), 20
column_power (module), 21
column_scale (module), 22
column_utils (module), 22
ColumnExponential (class in column_exponential), 18
ColumnLogarithm (class in column_logarithm), 19
ColumnOffset (class in column_offset), 20
ColumnPolynomial (class in column_polynomial), 20
ColumnPower (class in column_power), 21
ColumnScale (class in column_scale), 22
ColumnUtils (class in column_utils), 22
cubic_spline_resampler (module), 11
CubicSplineResampler (class in cubic_spline_resampler), 11

D

dedup_wavelengths() (reader_utils.ReaderUtils method), 10
del_data() (sdal_spectrum.SdalSpectrum method), 4
directory() (path_parts.PathParts method), 23

E

exponential() (column_exponential.ColumnExponential method), 18
extension() (path_parts.PathParts method), 23
extract_pieces() (splitter_utils.SplitterUtils method), 15

F

filename() (path_parts.PathParts method), 23
fill_mask() (sdal_collection.SdalCollection method), 7

G

get_collection() (sdal_dataset.SdalDataset method), 6
get_collection_names() (sdal_dataset.SdalDataset method), 6
get_data() (sdal_spectrum.SdalSpectrum method), 4
get_data_labels() (sdal_spectrum.SdalSpectrum method), 5
get_filepaths() (reader_utils.ReaderUtils method), 10
get_mask() (sdal_collection.SdalCollection method), 7
get_metadata() (sdal_spectrum.SdalSpectrum method), 5
get_metadata_labels() (sdal_spectrum.SdalSpectrum method), 5
get_name() (sdal_spectrum.SdalSpectrum method), 5
get_spectrum() (sdal_collection.SdalCollection method), 7
get_spectrums() (sdal_collection.SdalCollection method), 8
get_taps() (resampler_utils.ResamplerUtils method), 12

H

has_one_data_array() (spectrum_utils.SpectrumUtils method), 24

I

interval_of_wavelengths_splitter (module), 14
IntervalOfWavelengthsSplitter (class in interval_of_wavelengths_splitter), 14
is_label_valid() (spectrum_utils.SpectrumUtils method), 24
is_meta_label_valid() (spectrum_utils.SpectrumUtils method), 24
is_piece_index_valid() (spectrum_utils.SpectrumUtils method), 24
is_piece_order_valid() (spectrum_utils.SpectrumUtils method), 24
is_valid() (column_utils.ColumnUtils method), 22
is_valid_collection() (sdal_dataset.SdalDataset method), 7
is_valid_data_label() (sdal_spectrum.SdalSpectrum method), 5
is_valid_filepath() (reader_utils.ReaderUtils method), 10
is_valid_meta_label() (sdal_spectrum.SdalSpectrum method), 5

J

jump_correcting_stitcher (module), 17
JumpCorrectingStitcher (class in jump_correcting_stitcher), 17

L

linear_resampler (module), 11
LinearResampler (class in linear_resampler), 11
load_spectrum() (asd_reader.AsdReader method), 8
load_spectrum() (sed_reader.SedReader method), 8
load_spectrum() (sig_reader.SigReader method), 9
logarithm() (column_logarithm.ColumnLogarithm method), 19

N

nondecreasing_wavelengths_splitter (module), 13
NondecreasingWavelengthsSplitter (class in nondecreasing_wavelengths_splitter), 13

O

offset() (column_offset.ColumnOffset method), 20
overlap_averaging_stitcher (module), 16
overlap_blending_stitcher (module), 16
OverlapAveragingStitcher (class in overlap_averaging_stitcher), 16
OverlapBlendingStitcher (class in overlap_blending_stitcher), 16

P

parts() (path_parts.PathParts method), 23
path_parts (module), 23
PathParts (class in path_parts), 23
polynomial() (column_polynomial.ColumnPolynomial method), 20
power() (column_power.ColumnPower method), 21
print_data() (sdal_spectrum.SdalSpectrum method), 5
print_metadata() (sdal_spectrum.SdalSpectrum method), 6
print_summary() (sdal_collection.SdalCollection method), 8
print_summary() (sdal_dataset.SdalDataset method), 7

R

reader_utils (module), 10
ReaderUtils (class in reader_utils), 10
resample() (cubic_spline_resampler.CubicSplineResampler method), 11
resample() (linear_resampler.LinearResampler method), 11
resampler_utils (module), 12
ResamplerUtils (class in resampler_utils), 12
run() (asd_reader.AsdReader method), 8
run() (at_specified_wavelengths_splitter.AtSpecifiedWavelengthsSplitter method), 13
run() (closest_approach_stitcher.ClosestApproachStitcher method), 18
run() (column_exponential.ColumnExponential method), 19
run() (column_logarithm.ColumnLogarithm method), 19
run() (column_offset.ColumnOffset method), 20
run() (column_polynomial.ColumnPolynomial method), 21
run() (column_power.ColumnPower method), 21
run() (column_scale.ColumnScale method), 22
run() (cubic_spline_resampler.CubicSplineResampler method), 12
run() (interval_of_wavelengths_splitter.IntervalOfWavelengthsSplitter method), 14
run() (jump_correcting_stitcher.JumpCorrectingStitcher method), 17
run() (linear_resampler.LinearResampler method), 11
run() (nondecreasing_wavelengths_splitter.NondecreasingWavelengthsSplit method), 13
run() (overlap_averaging_stitcher.OverlapAveragingStitcher method), 16
run() (overlap_blending_stitcher.OverlapBlendingStitcher method), 16
run() (sed_reader.SedReader method), 9
run() (sig_reader.SigReader method), 9

S

scale() (column_scale.ColumnScale method), 22
sdal_collection (module), 7
sdal_dataset (module), 6
sdal_spectrum (module), 4
sdal_utils (module), 23

SdalCollection (class in sdal_collection), 7
SdalDataset (class in sdal_dataset), 6
SdalSpectrum (class in sdal_spectrum), 4
SdalUtils (class in sdal_utils), 23
sed_reader (module), 8
SedReader (class in sed_reader), 8
set_name() (sdal_spectrum.SdalSpectrum method), 6
sig_reader (module), 9
SigReader (class in sig_reader), 9
spectrum_utils (module), 24
SpectrumUtils (class in spectrum_utils), 24
split() (at_specified_wavelengths_splitter.AtSpecifiedWavelengthsSplitter method), 14
split() (interval_of_wavelengths_splitter.IntervalOfWavelengthsSplitter method), 14
split() (nondecreasing_wavelengths_splitter.NondecreasingWavelengthsSplitter method), 13
splitter_utils (module), 15
SplitterUtils (class in splitter_utils), 15
stitch() (closest_approach_stitcher.ClosestApproachStitcher method), 18
stitch() (jump_correcting_stitcher.JumpCorrectingStitcher method), 17
stitch() (overlap_averaging_stitcher.OverlapAveragingStitcher method), 16
stitch() (overlap_blending_stitcher.OverlapBlendingStitcher method), 17

U

update_mask() (sdal_collection.SdalCollection method), 8
update_mask() (sdal_dataset.SdalDataset method), 7