sentinelSimulator Documentation

Release 0.0.1

E. Pinnington

August 07, 2017

CONTENTS

CHAPTER

ONE

CONTENTS:

1.1 sentinelSimulator

This Python package simulates Sentinel 2 data based on output from the JULES land surface model. Here we provide documentation on this package. Below we illustrate a basic explanation of the package in use.

Example of using sentinelSimulator package:

```
import simulator
# Get your stuff done
simulator.do_example()
```

1.1.1 Features

- Simulates sentinel 2 data from JULES output
- Add additional explanation

1.1.2 Source Code

github.com/example_user/sentinelSimulator

1.1.3 Support

If you are having issues, please let us know. Contact: ewan.pinnington@gmail.com

1.1.4 License

Details of licensing information.

1.2 sentinel_simulator

1.2.1 sentinel_simulator package

Subpackages

sentinel_simulator.jules package

Submodules

sentinel_simulator.jules.py_importNML module

`sentinel_simulator.jules.py_importNML.`**`importJulesNML`**`(`*nml*`)`

Parse a JULES nml file and write it in the style used for the julesNML class.

Parameters **nml** (*str.*) – JULES NML file name.

Returns None

sentinel_simulator.jules.py_jules module

`sentinel_simulator.jules.py_jules.`**`crop_run`**`(`*sow_date=110*, *b=6.631*, *smwilt=0.1866*, *neff=0.00057*`)`

Function that runs JULES with crop model turned on and given user defined parameters at Wallerfing site. Output is saved in folder and file specified within function.

Parameters

- **sow_date** (*int.*) – Sow date, between 90 and 150.
- **b** (*float.*) – Brooks-Corey exponent factor.
- **smwilt** (*float.*) – Soil moisture wilting point.
- **neff** (*float.*) – Nitrogen use efficiency of crop (Vcmax).

Returns 'Done' to notify used JULES run has finished.

Return type str

class `sentinel_simulator.jules.py_jules.`**`jules`**`(`*jules_exe='/home/if910917/jules/models/jules4.8/build/bin/jules.exe'*`)`

Bases: `sentinel_simulator.jules.py_jules.julesAllNML`

Class to run JULES.

Parameters **jules_exe** (*str*) – location of JULES executable.

Note: You must have JULES installed on local system with a version of 4.8 or higher.

`runJules``()`

Write all NML files to disk. Run JULES in a subprocess. Check output for fatal errors.

Returns stdout and stderr output from JULES model run.

Return type str

class `sentinel_simulator.jules.py_jules.`**`julesAllNML`**

This class is populated by the contents of a module which contains templates of all the required JULES namelist files

`writeNML``()`

sentinel_simulator.jules.py_julesNML module This module holds JULES namelist files. It has been automatically generated.

class `sentinel_simulator.jules.py_julesNML.`**`julesNML`**`(`*template*, *filename*`)`

This is the base class for storing and writing JULES namelist files

`update``(`*template*`)`

`write``()`

Module contents

Submodules

sentinel_simulator.opticalCanopyRT module

`sentinel_simulator.opticalCanopyRT.`**`canopyRTOptical`**`(`*state*, *geom*, *resln=1.0*`)`

A python wrapper to the SemiDiscrete optical canopy RT model of Nadine Gobron. Runs the model for the the whole of its valid spectra range at a resolution set by resln.

Parameters

- **state** (*instance.*) – Instance of the stateVector class.
- **geom** (*instance.*) – Instance of the sensorGeomety class.
- **resln** (*float.*) – the spectral resolution in nm [optional].

Returns Instance of the spectra class.

Return type instance.

sentinel_simulator.satelliteGeometry module

`sentinel_simulator.satelliteGeometry.`**`getSentinel2Geometry`**`(`*startDateUTC*, *lengthDays*, *lat*, *lon*, *alt=0.0*, *mission='Sentinel-2a'*, *tleFile='./TLE/norad_resource_tle.txt'*`)`

Calculate approximate geometry for Sentinel overpasses. Approximate because it assumes maximum satellite elevation is the time at which target is imaged.

Parameters

- **startDateUTC** (*object*) – a datetime object specifying when to start prediction.
- **lengthDays** (*int*) – number of days over which to perform calculations.
- **lat** (*float*) – latitude of target.
- **lon** (*float*) – longitude of target.
- **alt** (*float*) – altitude of target (in km).
- **mission** (*str*) – mission name as in TLE file.
- **tleFile** (*str*) – TLE file.

Returns a python list containing instances of the sensorGeometry class arranged in date order.

Return type list

class `sentinel_simulator.satelliteGeometry.`**`sensorGeometry`**

Class to hold sun-sensor geometry information.

`printGeom``()`

Prints currently specified class attributes.

sentinel_simulator.spectra module

exception `sentinel_simulator.spectra.`**`UnknownFileType`**

Bases: `exceptions.Exception`

Exception class for unknown filetypes

`sentinel_simulator.spectra.`**`convolve`**`(`*s1orig*, *s2orig*, *resln=1.0*, *s2norm=True*`)`

Convolve one spectra with another, for example to apply a band pass, or a spectral response function.

Parameters

- **s1orig** (*object*) – A spectra object.
- **s2orig** (*float*) – A spectra object.
- **resln** (*float*) – The spectral resolution to use.
- **s2norm** (*bool*) – If True normalise the second spectra (e.g. to apply a spectra response function).

Returns Convolved spectra.

Return type object

`sentinel_simulator.spectra.`**`sentinel2`**`(`*s*, *mission='a'*`)`

class `sentinel_simulator.spectra.`**`spectra`**`(`*fname=None*, *ftype='SVC'*, *wavlCol=0*, *reflCol=1*, *hdrLines=1*`)`

Bases: `object`

Spectra class for sentinel simulator.

`interpolate``(`*resltn=0.1*`)`

Interpolate spectra to the given resolution. Overwites exisiting data.

Parameters **resltn** (*float*) – resolution of the interpolation.

`loadCSV``(`*f*, *wavlCol=0*, *reflCol=1*, *hdrLines=1*`)`

Read in data from a standard CSV file object.

Parameters

- **f** (*file*) – File object.
- **wavlCol** (*int*) – Column containing wavelengths.
- **reflCol** (*int*) – Column containing reflectance data.
- **hdrLines** (*int*) – Number of lines to skip at start of file.

`loadSVCSig``(`*f*`)`

Read in data from an SVC .sig ascii file.

Parameters **f** (*file*) – File object.

`loadSpectra``(`*fname*, *wavlCol=0*, *reflCol=1*, *hdrLines=1*`)`

Load in the spectra from a given file using a method appropriate to the type of file.

Note: Current supported formats are:

SVC - SCV .sig ascii file

CSV - standard ascii comma seperated values

Parameters

- **fname** (*str*) – Valid filename containing spectra.
- **wavlCol** (*int*) – Column containing wavelengths.
- **reflCol** (*int*) – Column containing reflectance data.
- **hdrLines** (*int*) – Number of lines to skip at start of file.

`trim`(*wlmin*, *wlmax*)

Trim the spectra so it is between two specified wavelengths. Destroys the original data.

Parameters

- **wlmin** (*float*) – The lowest wavelength of the new spectra.
- **wlmax** (*float*) – The highest wavelength of the new spectra.

sentinel_simulator.stateVector module

`sentinel_simulator.stateVector.read`(*file_format='jules'*, *file_str=None*, *year=None*)

Reads output data to a dictionary of state vectors indexed by time.

Note: This function requires sub-functions capable of reading specified file format.

Parameters

- **file_format** – format of output to read.
- **file_str** (*str*) – location of file.
- **year** (*int*) – year of data to extract, if equal to None whole time series extracted

Returns state dictionary.

Return type dict

`sentinel_simulator.stateVector.read_jules`(*nc_file=None*, *year=None*)

Reads jules output from netCDF file and writes it to a dictionary indexed by date.

Parameters

- **nc_file** (*str*) – location of nc_file.
- **year** (*int*) – year of data to extract, if equal to None whole time series extracted.

Returns state dictionary.

Return type dict

class `sentinel_simulator.stateVector.stateVector`

Class to hold state vector data for optical and microwave canopy RT models.

Module contents

CHAPTER

TWO

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*

PYTHON MODULE INDEX