pysagereader Documentation

Release 0.2.0

Landon Rieger

Sep 22, 2021

Contents:

pysagereader is a python reader for the SAGE II data. Data is imported in either an `xarray` data structure or a dictionary of numpy arrays.

CHAPTER 1

Quickstart

1.1 Get the data

The SAGE II data is not provided with the pysagereader package but can be obtained from NASA ASDC.

1.2 Install the package

To install the package from PyPI

```
pip install pysagereader
```

1.3 Loading the data

By default data is loaded into an xarray dataset that can be easily sliced, subset, and plotted.

```
from pysagereader import SAGEIILoaderV700
sage = SAGEIILoaderV700(data_folder=r'/path/to/sage/data')
data = sage.load_data(min_date='2000-1-1', max_date='2003-12-31', min_lat=-10, max_
↪lat=10)
data.O3.plot(x='time', robust=True)
```

1.4 Creating NetCDF files

The xarray package also provides convenient export to NetCDF files

```
from pysagereader import SAGEIILoaderV700
sage = SAGEIILoaderV700(data_folder=r'/path/to/sage/data')
data = sage.load_data(min_date='2000-1-1', max_date='2001-12-31')
data.to_netcdf('sage_ii_v700_2000.nc')
```

Or from the command line:

```
python /install/directory/pysagereader/make_netcdf.py -i /sageii/data/folder -o /
↪output/folder -time_res yearly
```

CHAPTER 2

SAGE II Loader

class pysagereader.**SAGEIILoaderV700**(*data_folder: str = None*, *output_format: str = 'xarray'*, *species: List[str] = ('aerosol', 'h2o', 'no2', 'ozone', 'background')*, *cf_names: bool = False*, *filter_aerosol: bool = False*, *filter_ozone: bool = False*, *enumerate_flags: bool = False*, *normalize_percent_error: bool = False*, *return_separate_flags: bool = False*)

Class designed to load the v7.00 SAGE II spec and index files provided by NASA ADSC into python

Data files must be accessible by the users machine, and can be downloaded from: https://eosweb.larc.nasa.gov/project/sage2/sage2_v7_table

Parameters

- **data_folder** – location of sage ii index and spec files.
- **output_format** – format for the output data. If `'xarray'` the output is returned as an `xarray.Dataset`. If None the output is returned as a dictionary of numpy arrays.

  NOTE: the following options only apply to xarray output types
- **species** – Species to be returned in the output data. If None all species are returned. Options are `aerosol`, `ozone`, `h2o`, and `no2`. If more than one species is returned fields will be NaN-padded where data is not available. `species` is only used if `'xarray'` is set as the `output_data` format, otherwise it has no effect.
- **cf_names** – If True then CF-1.7 naming conventions are used for the output_data when `xarray` is selected.
- **filter_aerosol** – filter the aerosol using the cloud flag
- **filter_ozone** – filter the ozone using the criteria recommended in the release notes
  - Exclusion of all data points with an uncertainty estimate of 300% or greater
  - Exclusion of all profiles with an uncertainty greater than 10% between 30 and 50 km
  - Exclusion of all data points at altitude and below the occurrence of an aerosol extinction value of greater than 0.006 km^-1

  - Exclusion of all data points at altitude and below the occurrence of both the 525nm aerosol extinction value exceeding 0.001 km^-1 and the 525/1020 extinction ratio falling below 1.4
  - Exclusion of all data points below 35km an 200% or larger uncertainty estimate
- **enumerate_flags** – expand the index and species flags to their boolean values.
- **normalize_percent_error** – give the species error as percent rather than percent * 100
- **return_separate_flags** – return the enumerated flags as a separate data array

Example

```
>>> sage = SAGEIILoaderV700()
>>> sage.data_folder = 'path/to/data'
>>> data = sage.load_data('2004-1-1','2004-5-1')
```

In addition to the sage ii fields reported in the files, two additional time fields are provided to allow for easier subsetting of the data.

`data['mjd']` is a numpy array containing the modified julian dates of each scan

`date['time']` is an pandas time series object containing the times of each scan

static convert_index_bit_flags(*data: Dict[KT, VT]*) → xarray.core.dataset.Dataset
Convert the int32 index flags to a dataset of distinct flags

Parameters **data** – Dictionary of input data as returned by `load_data`

Returns

Return type Dataset of the index bit flags

static convert_species_bit_flags(*data: Dict[KT, VT]*) → xarray.core.dataset.Dataset
Convert the int32 species flags to a dataset of distinct flags

Parameters **data** – Dictionary of input data as returned by *load_data*

Returns

Return type Dataset of the index bit flags

convert_to_xarray(*data: Dict[KT, VT]*) → Union[xarray.core.dataset.Dataset, Tuple[xarray.core.dataset.Dataset, xarray.core.dataset.Dataset]]

Parameters **data** – Data from the `load_data` function

Returns

Return type data formatted to an xarray Dataset

get_index_filename(*year: int*, *month: int*) → str
Returns the index filename given a year and month

Parameters

- **year** – year of the data that will be loaded
- **month** – month of the data that will be loaded

Returns

Return type filename of the index file where the data is stored

static get_index_format() → Dict[str, Tuple[str, int]]
index format taken from sg2_indexinfo.pro provided in the v7.00 download

used for reading the binary data format

Returns an ordered dictionary of variables provided in the index file. Each dictionary field contains a tuple with the information (data type, length). Ordering is important as the sage ii binary files are read sequentially.

Return type Dict

get_spec_filename(*year: int*, *month: int*) → str
Returns the spec filename given a year and month

Parameters

- **year** – year of the data that will be loaded
- **month** – month of the data that will be loaded

Returns

Return type filename of the spec file where the data is stored

static get_spec_format() → Dict[str, Tuple[str, int]]
spec format taken from sg2_specinfo.pro provided in the v7.00 download

used for reading the binary data format

Returns Ordered dictionary of variables provided in the spec file. Each dictionary field contains a tuple with the information (data type, number of data points). Ordering is important as the sage ii binary files are read sequentially.

Return type Dict

load_data(*min_date: str*, *max_date: str*, *min_lat: float = -90*, *max_lat: float = 90*, *min_lon: float = -180*, *max_lon: float = 360*) → Union[Dict[KT, VT], xarray.core.dataset.Dataset]
Load the SAGE II data for the specified dates and locations.

Parameters

- **min_date** – start date where data will be loaded in iso format, eg: '2004-1-1'
- **max_date** – end date where data will be loaded in iso format, eg: '2004-1-1'
- **min_lat** – minimum latitude (optional)
- **max_lat** – maximum latitude (optional)
- **min_lon** – minimum longitude (optional)
- **max_lon** – maximum longitude (optional)

Returns

Return type Variables are returned as numpy arrays (1 or 2 dimensional depending on the variable)

read_index_file(*file: str*) → Dict[KT, VT]
Read the binary file into a python data structure

Parameters **file** – filename to be read

Returns

Return type data from the file

read_spec_file(*file: str*, *num_profiles: int*) → List[Dict[KT, VT]]

Parameters

- **file** – name of the spec file to be read
- **num_profiles** – number of profiles to read from the spec file (usually determined from the index file)

Returns

Return type list of dictionaries containing the spec data. Each list is one event

static subset_data(*data: Dict[KT, VT], min_date: str, max_date: str, min_lat: float, max_lat: float, min_lon: float, max_lon: float*) → Dict[KT, VT]

Removes any data from the dictionary that does not meet the specified time, latitude and longitude requirements.

Parameters

- **data** – dictionary of sage ii data. Must have the fields 'mjd', 'Lat' and 'Lon'. All others are optional
- **min_date** – start date where data will be loaded in iso format, eg: '2004-1-1'
- **max_date** – end date where data will be loaded in iso format, eg: '2004-1-1'
- **min_lat** – minimum latitude (optional)
- **max_lat** – maximum latitude (optional)
- **min_lon** – minimum longitude (optional)
- **max_lon** – maximum longitude (optional)

Returns

Return type returns the dictionary with only data in the valid latitude, longitude and time range

CHAPTER 3

SAGE II

3.1 Variables

As far as possible the original SAGE II variable names used in the IDL scripts and documentation have been adopted. If *xarray* is chosen as the output format revision and creation date information is not included in the output structure.

3.1.1 Altitude Range for Species

Species	Range (km)
Ozone	5-60
NO2	15-60
Aerosol	1-45
Water Vapor	MSL-40

3.1.2 Revision Info

Field	Description
Num_Prof	Number of profiles (records) in file
Met_Rev_Date	LaRC Met Model Rev Date (yyyymmdd)
Driver_Rev	LaRC Driver version (eg. 6.20)
Transmission_Rev	LaRC Transmission version
Inversion_Rev	LaRC Inversion version
Spectroscopy_Rev	LaRC Inversion version
Eph_File_Name	Ephemeris file name
Met_File_Name	Met file name
Ref_File_Name	Refraction file name
Trans_File_Name	Transmission file name
Spec_File_Name	Species profile file name
FillVal	Fill value

3.1.3 Altitude grid and range info

Grid_Size	Altitude Grid spacing
Alt_Grid	Geometric Alt
Alt_Mid_Atm	Geometric Alt for Dens_Mid_Atm
Range_Trans	Transmission Min & Max alt
Range_O3	Ozone Density Min & Max alt
Range_NO2	NO2 Density Min & Max alt
Range_H2O	H2O Density Min & Max alt
Range_Ext	Extinction Min & Max alt
Range_Density	Density Min & Max alt
Range_Surface	Surface Area Min & Max alt

3.1.4 Event Specific Info

YYYYMMDD	Event Date (yyyymmdd) at 30 km
event_num	The event number
HHMMSS	Event Time (hhmmss) at 30 km
Day_Frac	Time of Year (ddd.fraction)
Lat	Sub-tangent Lat at 30km
Lon	Sub-tangent Lon at 30km
Beta	Spacecraft Beta angle (degree
Duration	Duration of event (seconds)
Type_Sat	Instrument Event Type, 0=sr, 1=ss)
Type_Tan	Event Type, Local (0=sr,1=ss)

3.1.5 Process Tracking Flag info

Processing Success	
Dropped	Value is non-zero if event is dropped
InfVec	32 bits describing the event processing
Ephemeris	
Eph_Cre_Date	Record creation date (yyyymmdd)
Eph_Cre_Time	Record creation time (hhmmss)
Met	
Met_Cre_Date	Record creation date (yyyymmdd)
Met_Cre_Time	Record creation time (hhmmss)
Refraction	
Ref_Cre_Date	Record creation date (yyyymmdd)
Ref_Cre_Time	Record creation time (hhmmss)
Transmission	
TRANS_Cre_Date	Record creation date (yyyymmdd)
TRANS_Cre_Time	Record creation time (hhmmss)
Inversion	
SPECIES_Cre_Date	Record creation date (yyyymmdd)
SPECIES_Cre_Time	Record creation time (hhmmss)

3.1.6 Species File Contents

Field Type Description

Tan_Alt	Center-of-Sun Tangent Alt (km)
Tan_Lat	Center-of-Sun Lat (deg)
Tan_Lon	Center-of-Sun Lon (deg)
NMC_Pres	Pressure (mb) (0.5-70km)
NMC_Temp	Temperature (K), (0.5-70km)
NMC_Dens	Density (molecules/cm3) (.5-70km)
NMC_Dens_Err	Density Uncertainty(%x100)
Trop_Height	Tropopause height in km
Wavelength	Channel wavelengths
O3	O3 number density (cm-3)
NO2	NO2 number density (cm-3)
H2O	H2O number density (ppp)
Ext386	386 nm aerosol extinction (1/km)
Ext452	452 nm aerosol extinction (1/km)
Ext525	525 nm aerosol extinction (1/km)
Ext1020	1020 nm aerosol extinction (1/km)
Density	Molecular density (1/cm^3)
SurfDen	Aerosol surface area density (micrometer^2/cm^3)
Radius	Aerosol effective radius (micrometer)
Dens_Mid_Atm	Middle atmosphere retrieved density(1/cm^3)
O3_Err	o3 number density uncertainty (%x100)
NO2_Err	NO2 number density uncertainty (%x100)
H2O_Err	H2O number density uncertainty (%x100)
Ext386_Err	386 nm aerosol ext. uncertainty (%x100)

Continued on next page

Table 1 – continued from previous page

Ext452_Err	452 nm aerosol ext. uncertainty (%x100)
Ext525_Err	525 nm aerosol ext. uncertainty (%x100)
Ext1020_Err	1020 nm aerosol ext. uncertainty (%x100)
Density_Err	Density uncertainty (%x100)
SurfDen_Err	Aerosol surface area density uncertainty(%x100)
Radius_Err	Aerosol effective radius uncertainty (%x100)
Dens_Mid_Atm_Err	Middle atmosphere density uncertainty (%x100)
InfVec	Bit-wise quality flags

3.2 Quality Flags

SAGE II data returns both event (index) flags as well as species flags. These are 32 bit integers contained in the *InfVec* and *ProfileInfVec* variables respectively. However, for easier use the flags can be expanded to show each bit separately.

```
from pysagereader.sage_ii_reader import SAGEIILoaderV700

sage = SAGEIILoaderV700(data_folder=r'path\to\sage\data', enumerate_flags=True)
data = sage.load_data('2000-1-1', '2003-12-31', -10, 10)
```

The flags can also be returned in a separate array for convenience.

```
from pysagereader.sage_ii_reader import SAGEIILoaderV700

sage = SAGEIILoaderV700(data_folder=r'path\to\sage\data', return_separate_flags=True)
data, flags = sage.load_data('2000-1-1', '2003-12-31', -10, 10)
```

3.3 Data Filtering

3.3.1 Ozone

It is recommend that only a subset of the ozone data be used for scientific analysis, based on filtering recommendations from the release notes. Ozone results that meet these criteria can be determined from the `ozone_filter` variable in the returned dataset. A value of `0` indicates ozone should not be used. The following criteria are used as the filters:

- Exclusion of all data points with an uncertainty estimate of 300% or greater
- Exclusion of all profiles with an uncertainty greater than 10% between 30 and 50 km
- Exclusion of all data points at altitude and below the occurrence of an aerosol extinction value of greater than 0.006 km^{-1}
- Exclusion of all data points at altitude and below the occurrence of both the 525nm aerosol extinction value exceeding 0.001 km^{-1} and the 525/1020 extinction ratio falling below 1.4
- Exclusion of all data points below 35km an 200% or larger uncertainty estimate

3.3.2 Aerosol

To remove cloud contamination from the aerosol data flags `Cloud_Bit_1` and `Cloud_Bit_2` are used to compute the `cloud_filter`. A value of `1` indicates there is a cloud present at or above that altitude.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index