smos Documentation

Release unknown

TU Wien

Jul 01, 2020

Contents

SMOS (Soil Moisture and Ocean Salinity) data readers and time series converter.

Works great in combination with pytesmo.

CHAPTER 1

Documentation & Software Citation

To see the latest full documentation click on the docs badge at the top.

To cite this package follow the Zenodo badge at the top and export the citation there. Be aware that this badge links to the latest package version. Additional information on DOI versioning can be found here: http://help.zenodo.org/#versioning

CHAPTER 2

Installation

Before installing this package via pip, please install the necessary conda dependencies:

```
$ conda install -c conda-forge netcdf4 pyresample
$ pip install smos
```

Setup of a complete environment with conda can be performed using the following commands:

You can also install all needed (conda and pip) dependencies at once using the following commands after cloning this repository. This is recommended for developers of the package.

```
$ git clone https://github.com/TUW-GEO/smos.git --recursive
$ cd smos
$ conda create -n smos python=3.6 # or any supported python version
$ source activate smos
$ conda update -f environment.yml
$ python setup.py develop
```

or you can use the installation script with/without the develop flag (`-d`), which will call `python setup.py develop`, resp `python setup.py install`.

```
$ bash install.sh -d --python 3.6 --name smos
```

CHAPTER 3

Supported Products

Currently the following products are supported, additional products can be added.

- SMOS IC: SMOS INRA-CESBIO (SMOS-IC) 25km

CHAPTER 4

Contribute

We are happy if you want to contribute. Please raise an issue explaining what is missing or if you find a bug. We will also gladly accept pull requests against our master branch for new features or bug fixes.

4.1 Guidelines

If you want to contribute please follow these steps:

- Fork the smos repository to your account
- make a new feature branch from the smos master branch
- Add your feature
- please include tests for your contributions in one of the test directories We use py.test so a simple function called test_my_feature is enough
- submit a pull request to our master branch

CHAPTER 5

Reading images

5.1 L3_SMOS_IC

After downloading the data you will have a directory with subpaths of the format `YYYY`. Let's call this path `root_path`. To read 'Soil_Moisture' data for a certain date use the following code:

```python
from smos.smos_ic.interface import SMOSDs
import matplotlib.pyplot as plt
from datetime import datetime
import os
# make sure to clone testdata submodule from https://github.com/TUW-GEO/smos
from smos import testdata_path


root_path = os.path.join(testdata_path, 'L3_SMOS_IC', 'ASC')
ds = SMOSDs(root_path, parameters='Soil_Moisture')
image = ds.read(datetime(2018, 1, 1))
assert list(image.data.keys()) == ['Soil_Moisture']
sm_data = image.data['Soil_Moisture']

plt.imshow(sm_data)
plt.show()
```

The returned image is of the type pygeobase.Image. Which is only a small wrapper around a dictionary of numpy arrays.

If you only have a single image you can also read the data directly by specifying the file. Here we ignore any "Quality_Flag" values and simply read all data from file.

```python
from smos.smos_ic.interface import SMOSImg
import matplotlib.pyplot as plt
from datetime import datetime
import os
# make sure to clone testdata submodule from https://github.com/TUW-GEO/smos
```

(continues on next page)

(continued from previous page)

```
from smos import testdata_path

fname = os.path.join(testdata_path, 'L3_SMOS_IC', 'ASC', '2018',
                     'SM_RE06_MIR_CDF3SA_20180101T000000_20180101T235959_105_001_8.
↪DBL.nc')
img = SMOSImg(fname, read_flags=None)

image = img.read(datetime(2018,1,1))
sm_data = image.data['Soil_Moisture']

plt.imshow(sm_data)
plt.show()
```

You can also limit the reading to certain variables or a spatial subset by defining a bounding box area. In the following example, we read SMOS IC Soil Moisture over Europe only, and mask the data based on the “Quality_Flag” variable, to only include 0-flagged (i.e. “good”) values.

```
from smos.smos_ic.interface import SMOSImg
from smos.grid import EASE25CellGrid
import matplotlib.pyplot as plt
from datetime import datetime
import os
# make sure to clone testdata submodule from https://github.com/TUW-GEO/smos
from smos import testdata_path

fname = os.path.join(testdata_path, 'L3_SMOS_IC', 'ASC', '2018',
                     'SM_RE06_MIR_CDF3SA_20180101T000000_20180101T235959_105_001_8.
↪DBL.nc')

# bbox_order : (min_lon, min_lat, max_lon, max_lat)
subgrid_eu = EASE25CellGrid(bbox=(-11., 34., 43., 71.))

img = SMOSImg(fname, parameters=['Soil_Moisture', 'Quality_Flag', 'Days', 'UTC_Seconds
↪'],
              read_flags=(0,), grid=subgrid_eu)

imgdata = img.read(datetime(2018,1,1))

plt.imshow(imgdata.data['Soil_Moisture'])
plt.show()
```

CHAPTER 6

Write (subset) images

To write down an image to a new file (e.g. after filtering certain parameters or to create spatial subset files) you can use the functions `SMOSImg.write`. This will create a new netcdf file with the content of the current image at a selected location.

```
from smos.smos_ic.interface import SMOSImg
from smos.grid import EASE25CellGrid
import matplotlib.pyplot as plt
from datetime import datetime
import os
# make sure to clone testdata submodule from https://github.com/TUW-GEO/smos
from smos import testdata_path

fname = os.path.join(testdata_path, 'L3_SMOS_IC', 'ASC', '2018',
                     'SM_RE06_MIR_CDF3SA_20180101T000000_20180101T235959_105_001_8.
↪DBL.nc')

# bbox_order : (min_lon, min_lat, max_lon, max_lat)
subgrid_eu = EASE25CellGrid(bbox=(-11., 34., 43., 71.))

img = SMOSImg(fname, parameters=['Soil_Moisture', 'Quality_Flag', 'Days', 'UTC_Seconds
↪'],
              read_flags=(0,), grid=subgrid_eu)

img.read(datetime(2018,1,1))

img.write(r"C:\Temp\write\subset_image.nc")
```

Finally, you can also write down multiple files using the write function from `SMOSDs`. You can either create single files per time stamp (like the original data is) or netcdf stacks. This example will do both (note that days when no data is loaded are also skipped when writing the subset).

```
from smos.smos_ic.interface import SMOSDs
from smos.grid import EASE25CellGrid
import matplotlib.pyplot as plt
```

(continues on next page)

(continued from previous page)

```
from datetime import datetime
import os
# make sure to clone testdata submodule from https://github.com/TUW-GEO/smos
from smos import testdata_path

path = os.path.join(testdata_path, 'L3_SMOS_IC', 'ASC')

# bbox_order : (min_lon, min_lat, max_lon, max_lat)
subgrid_eu = EASE25CellGrid(bbox=(-11., 34., 43., 71.))

ds = SMOSDs(path, parameters=['Soil_Moisture', 'Quality_Flag', 'Days', 'UTC_Seconds'],
             read_flags=(0,), grid=subgrid_eu)

# write data as single files
ds.write_multiple(r'C:\Temp\write\test', start_date=datetime(2018,1,1), end_
↪date=datetime(2018,1,3),
                  stackfile=None)

# write data as a stack
ds.write_multiple(r'C:\Temp\write\test', stackfile='stack.nc', start_
↪date=datetime(2018,1,1),
                  end_date=datetime(2018,1,3))
```

CHAPTER 7

Variable naming for different versions of SMOS

This is a full list of all available variables in each image. These parameters can be passed to the repurpose routine to create time series.

- L3 SMOS IC

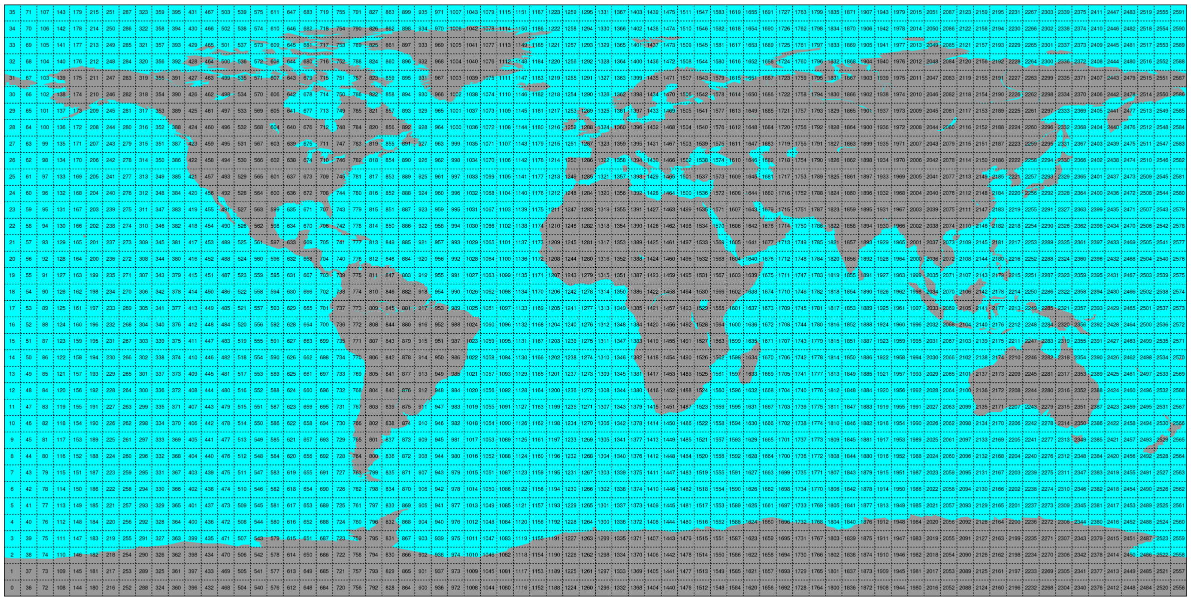
Parameter	Long Name	Units
Days	Number of Days since 1/1/2000	
Processing_Flags	Processing Flags	
Quality_Flag	0: data OK, 1: data not recommended, 2: missing data	
RMSE	RMSE TBmeas. TB modeled	
Scene_Flags	Scene Flags	
Soil_Moisture	Soil Moisture	$m^3 m^{-3}$
Soil_Moisture_StdError	Soil Moisture standard error	$m^3 m^{-3}$
Soil_Temperature_Level1	ECMWF Soil Temperature at surface level 1	Kelvin
UTC_Microseconds	Microseconds	micros.
UTC_Seconds	Number of Seconds	s

CHAPTER 8

Conversion to time series format

For a lot of applications it is favorable to convert the image based format into a format which is optimized for fast time series retrieval. This is what we often need for e.g. validation studies. This can be done by stacking the images into a netCDF file and choosing the correct chunk sizes or a lot of other methods. We have chosen to do it in the following way:

- Store the time series in netCDF4 in the Climate and Forecast convention Orthogonal multidimensional array representation
- Store the time series in 5x5 degree cells. This means there will be 2448 cell files for global data and a file called `grid.nc`, which contains the information about which grid point is stored in which file. This allows us to read a whole 5x5 degree area into memory and iterate over the time series quickly.



This conversion can be performed using the `smos_repurpose` command line program. An example would be:

```
smos_repurpose /smos_ic_img_data /timeseries/data 2011-01-01 2011-01-02 --parameters␣
↪Soil_Moisture --bbox -11 34 43 71
```

Which would take Soil_Moisture values from SMOS IC images stored in `/image_data` from January 1st 2011 to January 2nd 2011 and store the values as time series in the folder `/timeseries/data`.

Keywords that can be used in `smos_repurpose`:

- **-h (–help)** : Shows the help text for the reshuffle function
- **–parameters** : Parameters to reshuffle into time series format. e.g. Soil_Moisture. If this is not specified, all parameters in the first detected image file will be reshuffled. Default: None.
- **–only_good** : Read only 0-flagged (GOOD) observations (by Quality_Flag), if this is set to False, also 1-flagged (not recommended) ones will be read and reshuffled, 2-flagged (missing) values are always excluded. Excluded values are replaced by NaNs. Default: False.
- **–bbox** : min_lon min_lat max_lon max_lat. Bounding Box (lower left and upper right corner) of subset area of global images to reshuffle (WGS84). Default: None.
- **–imgbuffer** : The number of images that are read into memory before converting them into time series. Bigger numbers make the conversion faster but consume more memory. Default: 100.

Conversion to time series is performed by the repurpose package in the background. For custom settings or other options see the repurpose documentation .

CHAPTER 9

Contents

9.1 Reading images

9.1.1 L3_SMOS_IC

After downloading the data you will have a directory with subpaths of the format `YYYY`. Let's call this path `root_path`. To read 'Soil_Moisture' data for a certain date use the following code:

```python
from smos.smos_ic.interface import SMOSDs
import matplotlib.pyplot as plt
from datetime import datetime
import os
# make sure to clone testdata submodule from https://github.com/TUW-GEO/smos
from smos import testdata_path


root_path = os.path.join(testdata_path, 'L3_SMOS_IC', 'ASC')
ds = SMOSDs(root_path, parameters='Soil_Moisture')
image = ds.read(datetime(2018, 1, 1))
assert list(image.data.keys()) == ['Soil_Moisture']
sm_data = image.data['Soil_Moisture']

plt.imshow(sm_data)
plt.show()
```

The returned image is of the type pygeobase.Image. Which is only a small wrapper around a dictionary of numpy arrays.

If you only have a single image you can also read the data directly by specifying the file. Here we ignore any "Quality_Flag" values and simply read all data from file.

```python
from smos.smos_ic.interface import SMOSImg
import matplotlib.pyplot as plt
from datetime import datetime
```

(continues on next page)

(continued from previous page)

```
import os
# make sure to clone testdata submodule from https://github.com/TUW-GEO/smos
from smos import testdata_path

fname = os.path.join(testdata_path, 'L3_SMOS_IC', 'ASC', '2018',
                     'SM_RE06_MIR_CDF3SA_20180101T000000_20180101T235959_105_001_8.
↪DBL.nc')
img = SMOSImg(fname, read_flags=None)

image = img.read(datetime(2018,1,1))
sm_data = image.data['Soil_Moisture']

plt.imshow(sm_data)
plt.show()
```

You can also limit the reading to certain variables or a spatial subset by defining a bounding box area. In the following example, we read SMOS IC Soil Moisture over Europe only, and mask the data based on the "Quality_Flag" variable, to only include 0-flagged (i.e. "good") values.

```
from smos.smos_ic.interface import SMOSImg
from smos.grid import EASE25CellGrid
import matplotlib.pyplot as plt
from datetime import datetime
import os
# make sure to clone testdata submodule from https://github.com/TUW-GEO/smos
from smos import testdata_path

fname = os.path.join(testdata_path, 'L3_SMOS_IC', 'ASC', '2018',
                     'SM_RE06_MIR_CDF3SA_20180101T000000_20180101T235959_105_001_8.
↪DBL.nc')

# bbox_order : (min_lon, min_lat, max_lon, max_lat)
subgrid_eu = EASE25CellGrid(bbox=(-11., 34., 43., 71.))

img = SMOSImg(fname, parameters=['Soil_Moisture', 'Quality_Flag', 'Days', 'UTC_Seconds
↪'],
              read_flags=(0,), grid=subgrid_eu)

imgdata = img.read(datetime(2018,1,1))

plt.imshow(imgdata.data['Soil_Moisture'])
plt.show()
```

9.2 Write (subset) images

To write down an image to a new file (e.g. after filtering certain parameters or to create spatial subset files) you can use the functions `SMOSImg.write`. This will create a new netcdf file with the content of the current image at a selected location.

```
from smos.smos_ic.interface import SMOSImg
from smos.grid import EASE25CellGrid
import matplotlib.pyplot as plt
from datetime import datetime
import os
```

(continues on next page)

(continued from previous page)

```
# make sure to clone testdata submodule from https://github.com/TUW-GEO/smos
from smos import testdata_path

fname = os.path.join(testdata_path, 'L3_SMOS_IC', 'ASC', '2018',
                     'SM_RE06_MIR_CDF3SA_20180101T000000_20180101T235959_105_001_8.
↪DBL.nc')

# bbox_order : (min_lon, min_lat, max_lon, max_lat)
subgrid_eu = EASE25CellGrid(bbox=(-11., 34., 43., 71.))

img = SMOSImg(fname, parameters=['Soil_Moisture', 'Quality_Flag', 'Days', 'UTC_Seconds
↪'],
              read_flags=(0,), grid=subgrid_eu)

img.read(datetime(2018,1,1))

img.write(r"C:\Temp\write\subset_image.nc")
```

Finally, you can also write down multiple files using the write function from `SMOSDs`. You can either create single files per time stamp (like the original data is) or netcdf stacks. This example will do both (note that days when no data is loaded are also skipped when writing the subset).

```
from smos.smos_ic.interface import SMOSDs
from smos.grid import EASE25CellGrid
import matplotlib.pyplot as plt
from datetime import datetime
import os
# make sure to clone testdata submodule from https://github.com/TUW-GEO/smos
from smos import testdata_path

path = os.path.join(testdata_path, 'L3_SMOS_IC', 'ASC')

# bbox_order : (min_lon, min_lat, max_lon, max_lat)
subgrid_eu = EASE25CellGrid(bbox=(-11., 34., 43., 71.))

ds = SMOSDs(path, parameters=['Soil_Moisture', 'Quality_Flag', 'Days', 'UTC_Seconds'],
             read_flags=(0,), grid=subgrid_eu)

# write data as single files
ds.write_multiple(r'C:\Temp\write\test', start_date=datetime(2018,1,1), end_
↪date=datetime(2018,1,3),
                  stackfile=None)

# write data as a stack
ds.write_multiple(r'C:\Temp\write\test', stackfile='stack.nc', start_
↪date=datetime(2018,1,1),
                  end_date=datetime(2018,1,3))
```

9.3 Variable naming for different versions of SMOS

This is a full list of all available variables in each image. These parameters can be passed to the repurpose routine to create time series.

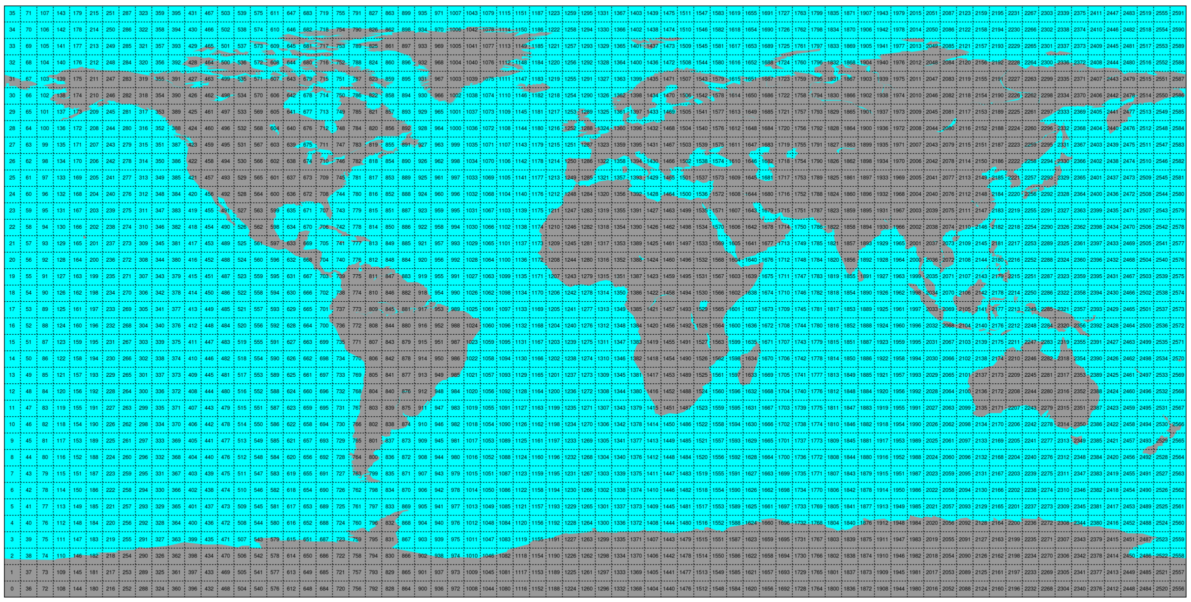
- L3 SMOS IC

Parameter	Long Name	Units
Days	Number of Days since 1/1/2000	
Processing_Flags	Processing Flags	
Quality_Flag	0: data OK, 1: data not recommended, 2: missing data	
RMSE	RMSE TBmeas. TB modeled	
Scene_Flags	Scene Flags	
Soil_Moisture	Soil Moisture	$m^3 m^{-3}$
Soil_Moisture_StdError	Soil Moisture standard error	$m^3 m^{-3}$
Soil_Temperature_Level1	ECMWF Soil Temperature at surface level 1	Kelvin
UTC_Microseconds	Microseconds	micros.
UTC_Seconds	Number of Seconds	s

9.4 Conversion to time series format

For a lot of applications it is favorable to convert the image based format into a format which is optimized for fast time series retrieval. This is what we often need for e.g. validation studies. This can be done by stacking the images into a netCDF file and choosing the correct chunk sizes or a lot of other methods. We have chosen to do it in the following way:

- Store the time series in netCDF4 in the Climate and Forecast convention Orthogonal multidimensional array representation
- Store the time series in 5x5 degree cells. This means there will be 2448 cell files for global data and a file called `grid.nc`, which contains the information about which grid point is stored in which file. This allows us to read a whole 5x5 degree area into memory and iterate over the time series quickly.



This conversion can be performed using the `smos_repurpose` command line program. An example would be:

```
smos_repurpose /smos_ic_img_data /timeseries/data 2011-01-01 2011-01-02 --parameters␣
↪Soil_Moisture --bbox -11 34 43 71
```

Which would take Soil_Moisture values from SMOS IC images stored in `/image_data` from January 1st 2011 to January 2nd 2011 and store the values as time series in the folder `/timeseries/data`.

Keywords that can be used in `smos_repurpose`:

- **-h (–help)** : Shows the help text for the reshuffle function
- **–parameters** : Parameters to reshuffle into time series format. e.g. Soil_Moisture. If this is not specified, all parameters in the first detected image file will be reshuffled. Default: None.
- **–only_good** : Read only 0-flagged (GOOD) observations (by Quality_Flag), if this is set to False, also 1-flagged (not recommended) ones will be read and reshuffled, 2-flagged (missing) values are always excluded. Excluded values are replaced by NaNs. Default: False.
- **–bbox** : min_lon min_lat max_lon max_lat. Bounding Box (lower left and upper right corner) of subset area of global images to reshuffle (WGS84). Default: None.
- **–imgbuffer** : The number of images that are read into memory before converting them into time series. Bigger numbers make the conversion faster but consume more memory. Default: 100.

Conversion to time series is performed by the repurpose package in the background. For custom settings or other options see the repurpose documentation .

9.5 License

9.6 Contributors

- Wolfgang Preimesberger <wolfgang.preimesberger@geo.tuwien.ac.at>

9.7 Changelog

9.7.1 Unreleased

-

9.7.2 Version 0.2

- Add option read data for spatial subsets
- Add option to add time stamps to time series index
- Add function to write images/stacks down
- Add function to reshuffle by bounding box
- Update docs and travis, automatic pypi deployment
- Update tests
- By default also reshuffle if Scene_Flag is 2
- Switch to pyscaffold 3 package structure
- Drop support for python 2

9.7.3 Version 0.1

- First version of L3 SMOS IC image and TS reader
- Include time series reshuffling
- Include tests and CI
- Basic documentation

9.8 smos

9.8.1 smos package

Subpackages

smos.smos_ic package

Submodules

smos.smos_ic.interface module

smos.smos_ic.reshuffle module

Module contents

Submodules

smos.grid module

Module contents

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index