MULTIPLY Prior Engine Documentation

Release 0.5.1

Joris Timmermans, Thomas Ramsauer

Nov 14, 2019

CONTENTS

CHAPTER

ONE

SCOPE OF MULTIPLY

The MULTIPLY project will “develop a new platform for joint and consistent retrieval of Copernicus SENTINEL data and beyond”.

This documentation covers the prior engine for the MULTIPLY main platform. This module provides *a priori* information to the Inference Engine to support land surface parameter retrieval.

The prior engine specific documentation is hosted on ReadTheDocs. It is part of the MULTIPLY core documentation. Please find the latest pdf version of this documentation here.

CHAPTER

TWO

FIRST STEPS

2.1 Getting Started

Please find instructions on how to download and install the prior engine in the *Installation* section.

Note: TBD: Getting started with python, bayes theorem, ..

2.2 Testing and Contribution

You are welcome to test and contribute to the MULTIPLY Prior Engine.

Please find corresponding guidelines and further information on how to do so in the *How to contribute* section and on the project GitHub page.

CHAPTER

THREE

CONTENT

3.1 Introduction

Priors are an essential component in the MULTIPLY inference engine as they provide a priori information on different components of the unknown state vector of the system, helping to constrain the ill-posed problem given that the information content from the observations alone is insufficient. A series of prior models with different levels of complexity is therefore required and will be developed and implemented as part of the MULTIPLY platform.

The priors to be implemented are:

- Differential characterisation of the traits of vegetation types or (crop) species
- Vegetation phenology
- Surface soil moisture dynamics
- Surface disturbances

3.1.1 Background

A seamless and gap free integration of SENTINEL data streams requires the transfer of information across temporal and spatial scales. Typically data gaps are filled using low pass filters and different interpolation techniques (e.g. Savitzky-Golay filter; Savitzky & Golay, 1964) directly on parameter space (e.g. Yuan et al., 2011; Kandasamy et al. 2013). However, this approach is inconsistent, as the ill-posed nature of the inversion problem results in strong correlations between parameters: smoothing one parameter breaks that relationship with other retrieved parameters. Additionally, the role of uncertainty is usually ignored in filtering. Given that filtering methods originate from a prior belief in the smoothness of the processes that control the evolution of the parameters, it makes sense to implement these smoothness constraints consistently as priors within the retrieval process. These so-called regularisation constraints encompass our prior belief in the spatial and temporal correlation of the parameter fields. These constraints are implemented within the MULTIPLY platform as a weak constraint. The added benefit of having these constraints is that they not only result in smoother and more consistent series (an added benefit is an important reduction in parameter uncertainty), but also in spatially and temporally gap free estimates of biophysical parameters.

However, other prior information should be used to better constrain the inversion, and make sure that the inferences on the parameters are consistent with our understanding of biogeochemical processes and their effect on the state of the land surface.

3.1.2 Goal

The major objectives of this software are i) to implement the required technical infrastructures to provide the prior information at appropriate temporal and spatial scales in relation to the SENTINEL observations, and ii) implement a flexible user interface which allows user to integrate own prior models as a MULTIPLY plugin.

3.2 Prior Data

3.2.1 Vegetation Prior Data

Note: TBD

3.2.2 Soil Moisture Prior Data

The provided prior data for the soil moisture domain is twofold. Mattia et al. [Mattia] show that the usage of climatological mean soil moisture information significantly improves soil moisture estimates from active microwave observations. Therefore, a soil moisture climatology is used as prior to get a general idea of the amplitude, variability and seasonal behaviour of the in situ soil moisture. Furthermore, a dynamic daily coarse resolution product is consulted for an a priori estimation of the current state.

The climatological prior data set has been generated from the global ESA CCI SM v04.4 COMBINED product which is derived from a combination of active and passive satellite sensors over the period 1978 - 2018. Originally, the data set provides daily surface soil moisture with a spatial resolution of 0.25 degree ([Dorigo]; [Gruber]; [Liu]). The data was aggregated to monthly means. Uncertainty is given by the intra-monthly standard deviation.

Data from the Soil Moisture Active Passive (SMAP) project is used as dynamic prior ([Reichle]). Specifically, the model-derived value-added Level 4 data product with 3-hourly estimates of soil moisture and respective error estimates at a 9 km resolution are averaged to daily values as the MULTIPLY platform assimilates data at this temporal resolution.

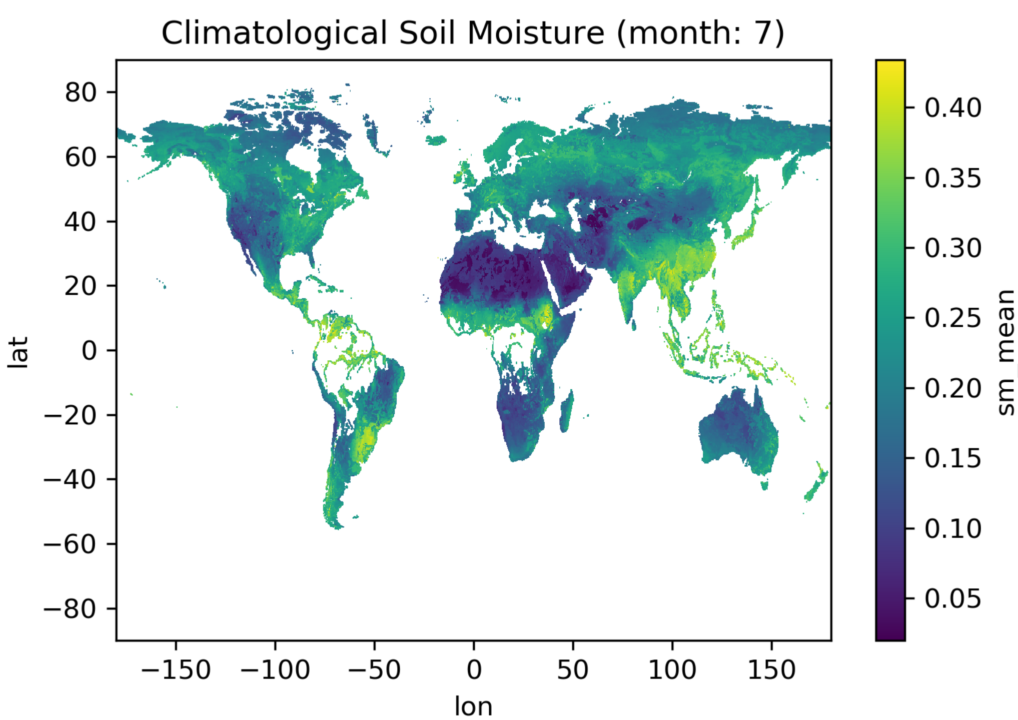

Fig. 1: Climatological Soil Moisture July

3.3 Installation

3.3.1 Download

If not already done so, the first step is to clone the latest code and change directory:

```
git clone https://github.com/multiply-org/prior-engine.git
cd prior-engine
```

Note: The MULTIPLY platform has been developed against Python 3.6. It cannot be guaranteed to work with previous Python versions.

3.3.2 Installation procedure

The MULTIPLY prior engine can be run from sources directly. To install the MULTIPLY prior engine into an existing Python environment just for the current user, use

```
python setup.py install --user
```

To install the MULTIPLY Core for development and for the current user, use

```
python setup.py develop --user
```

Using Conda

Note: TBD

3.3.3 Module requirements

from *requirements.txt*:

```
numpy==1.16
shapely==1.6
h5py==2.8
pandas==0.22
scipy==0.22
setuptools==40.8
matplotlib==2.2
pytest==4.6
gdal==2.4
netCDF4==1.5
PyYAML==3.12
typing
python_dateutil
recommonmark
```

3.4 Usage

3.4.1 Python Package

MULTIPLY prior engine is available as Python Package. To import it into your python application, use

```
import multiply_prior_engine
```

3.4.2 User defined priors

Users are provided the possibility to choose between prior-types, using the configuration file. This configuration file can be modified by both the users directly (using simple text editors), as well as the user-interface described below and in the upcoming MULTIPLY platform user-interface.

The user has three options to add prior data to the retrieval (in addition to choosing priors already made available by MULTIPLY).

- The user can choose to define single values for the prior in terms of transformed 'mu' and 'unc' values.
- The user can choose to provide a single geolocated tiff file, with both mean and uncertainty values. Here, the mean value should be provided as the first band, while the uncertainty of these values should be provided as the second band.
- Finally, the user can choose to provide a directory with multiple files, following a similar structure as the previous choice. Here, the files should be given a 8 digit date stamp in the filename.

The configuration file then could look like:

```
Prior
        General:
                directory_data: 'path 2 prior engine'
        LAI:
                database
                        static_dir: same as general directory_data
        SM:
                user:
                        mu: 0.5
                        unc: 0.02
        CWC:
                user:
                        file: 'path to geotiff-file'
        ALA:
                user:
                        dir: 'path to directory with geotiff-files (sorted on date)'

                ...


        output_directory: 'path to outputdirectory'
```

3.4.3 Command Line Interface

There is a Command Line Interface (CLI) integrated to allow for the following actions:

- add user defined prior data,

- import user defined prior data,
- remove/un-select prior data from configuration,
- show configuration.

The CLI's help can be accessed via *-h* flag:

```
user_prior -h
```

and will show:

```
usage: user_prior.py [-h] {show,S,add,A,remove,R,import,I} ...

Utility to integrate User Prior data in MULTIPLY Prior Engine

positional arguments:
{show,S,add,A,remove,R,import,I}
    show (S)            Show current prior config.
    add (A)             Add prior directory to configuration.
    remove (R)          Remove prior information from configuration.
    import (I)          Import user prior data.

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

The help and description of the above mentioned sub-commands can be accessed via, e.g.:

```
user_prior add -h
```

Note: If installed for the current user only, make sure the directory the prior engine gets installed to is in your PATH variable.

3.4.4 Logging

For now the Prior Engine has its own logging setup. To set the *logging level* please adjust the level accordingly in the *multiply_prior_engine/__init__.py* file. Available options are: NOTSET, DEBUG, INFO, WARNING, ERROR, CRITICAL.

3.5 Processing Flow

Priors are provided by the MULTIPLY prior engine for the respective forward operators. The relationships are shown in following figure:

Note: For information on **user defined prior files** please see the section on Usage.

3.5.1 Description of Prior Generation

This prototype is capable of delivering for both vegetation priors as well as soil priors spanning all variables required in the forward operators. The overall processing chain is divided up to two parts (dealing with the soil prior and the vegetation prior).

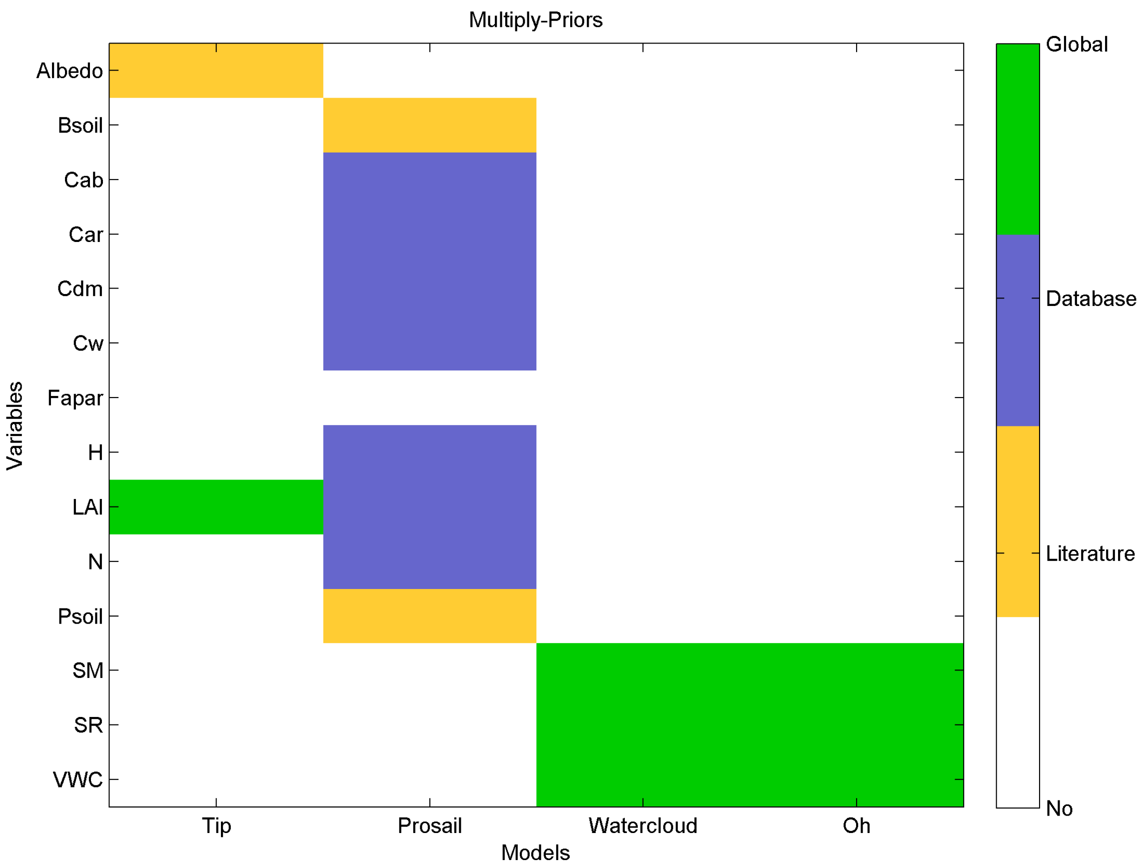

Fig. 2: Figure 1: Relationship of priors to their respective forward operators.

The optical prior engine is designed to deliver prior information to the inference engine specifically for the leaves and vegetation. The overall flow of the prior-engine is illustrated by Figure 2.

The 'microwave' prior engine is designed to deliver prior information for soil parameters. The overall flow of this part of the prior-engine is illustrated by Figure 3.

In these flowcharts a distinction is made between the current implementation of the prototype (green) and the final foreseen version of the prior engine (red). In order for completeness a place-holder (orange) process is embedded into the flowchart. In addition, in the final version of the prior engine the users themselves can choose between how the specific prior are used (see Usage). User-selections are obtained from the configuration-file with which the MULTIPLY framework is run. This is represented in the flowchart by orange selection boxes. Prior data specified by the User is currently not visualized for every prior generator.

Vegetation Priors

Within the prototype version of the module, the values of the priors are consistent with @peak biomass; no dynamical component is integrated into the prototype module.

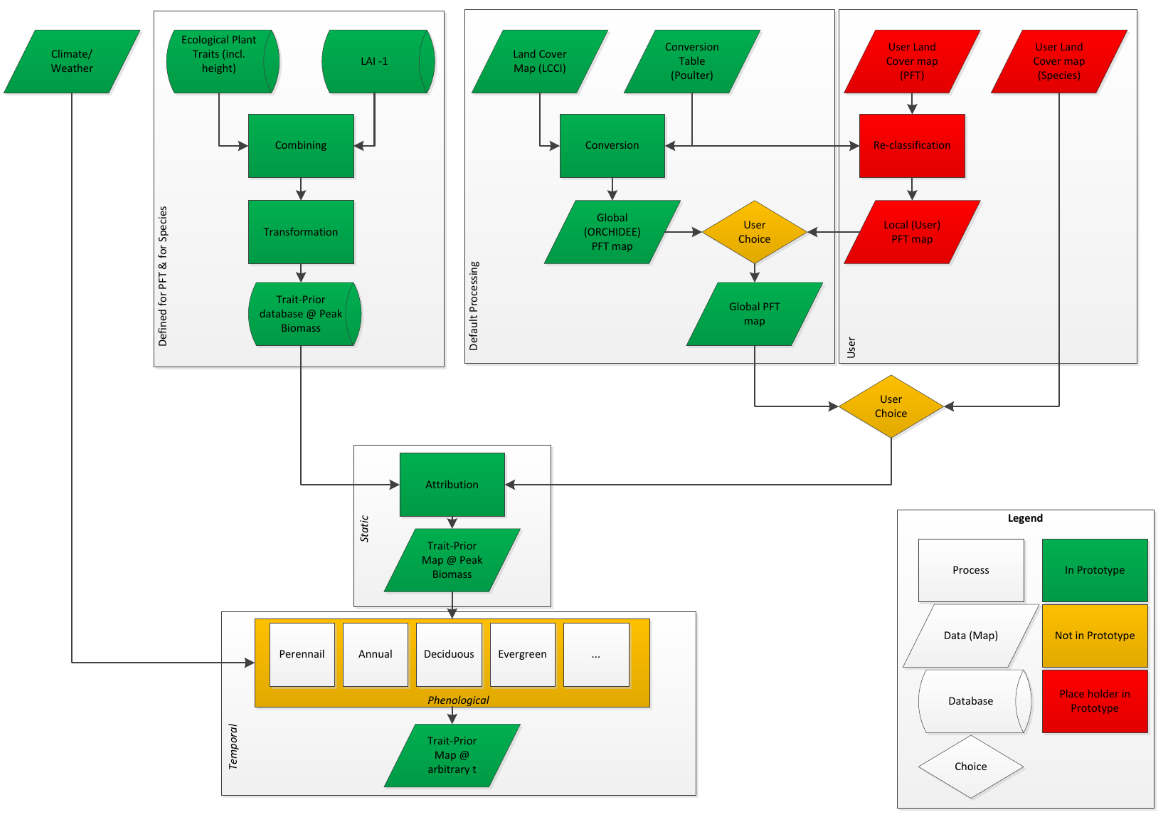

Fig. 3: Figure 2: Flow in 'optical' prior engine

Soil Priors

The included priors for soil moisture are currently twofold:

1) a climatological prior based on ESA CCI SM v04.4 data

2) a dynamic prior based on SMAP data

Please see the overall flow of this prior creator sub-engine below:

img/flow_prior_mic.png

Fig. 4: Figure 3: Flow in 'microwave' prior engine

Climatologic Priors

Mattia et al. [Mattia] show that the usage of climatological mean soil moisture information significantly improves soil moisture estimates from active microwave observations. Therefore, a soil moisture climatology is used as prior to get a general idea of the amplitude, variability and seasonal behaviour of the in situ soil moisture. Furthermore, a dynamic daily coarse resolution product is consulted for an a priori estimation of the current state.

The climatological prior data set has been generated from the global ESA CCI SM v04.4 COMBINED product which is derived from a combination of active and passive satellite sensors over the period 1978 - 2018 [GRUBER2019]. Originally, the data set provides daily surface soil moisture with a spatial resolution of 0.25 degree ([Dorigo]; [Gruber]; [Liu]). The data was aggregated to monthly means. Uncertainty is given by the intra-monthly standard deviation. There is also a interpolation routine included to allow for smooth inter monthly transitions.

img/clim_point.png

Fig. 5: Figure 4: Climatology soil moisture (bars) at point-scale with interpolated values (line)

Dynamic Priors

Data from the Soil Moisture Active Passive (SMAP) project is used as dynamic prior ([Reichle]). Specifically, the model-derived value-added Level 4 data product with 3-hourly estimates of soil moisture and respective error estimates at a 9 km resolution are averaged to daily values as the MULTIPLY platform assimilates data at this temporal resolution.

> "SMAP measurements provide direct sensing of soil moisture in the top 5 cm of the soil column. However, several of the key applications targeted by SMAP require knowledge of root zone soil moisture in the top 1 m of the soil column, which is not directly measured by SMAP. As part of its baseline mission, the SMAP project will produce model-derived value-added Level 4 data products to fill this gap and provide estimates of root zone soil moisture that are informed by and consistent with SMAP surface observations. Such estimates are obtained by merging SMAP observations with estimates from a land surface model in a data assimilation system. The land surface model component of the assimilation system is driven with observations-based meteorological forcing data, including precipitation, which is the most important driver for soil moisture. The model also encapsulates knowledge of key land surface processes, including the vertical transfer of soil moisture between the surface and root zone reservoirs. Finally, the model interpolates and extrapolates SMAP observations in time and in space, producing 3-hourly estimates of

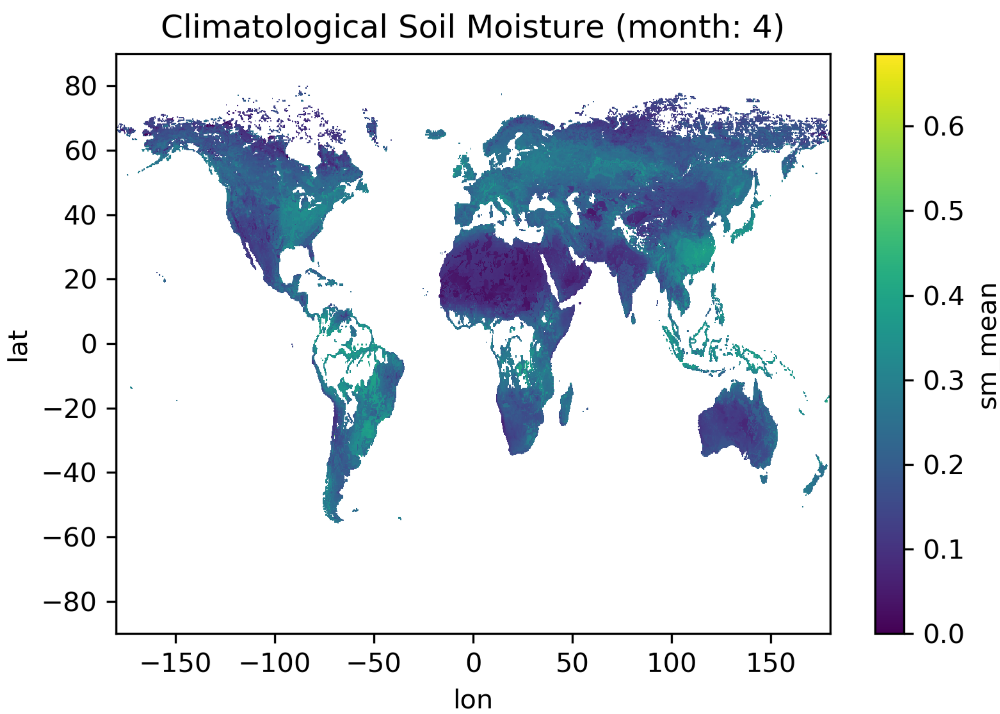

Fig. 6: Figure 5: Exemplary climatological soil moisture prior (mean) for April

soil moisture at a 9 km resolution. The SMAP L4_SM product thus provides a comprehensive and consistent picture of land surface hydrological conditions based on SMAP observations and complementary information from a variety of sources." [JPL]

The prior engine will rely on the MULTIPLY data-access component to download the appropriate data sets. These are then converted to be used by the inference engine. A valid registration on NASA's Earthdata Service is necessary.

3.5.2 Technical Description

The processing chain in the prior engine is defined in a config file. For now this looks like:

```
General:
    roi: POLYGON ((48.0 11.3, 48.2 11.300, 48.1 11.1, 48.0 11, 48.0 11.3))
    start_time: 2017-01-01
    end_time: 2017-12-31
    time_interval: 1 # 1 day
    spatial_resolution : 10 # metres
    state_mask: /path/to/my/state_mask.tif # Or shape?
    output_directory_root: /some/where/
    # output_prefix: my_test_33

Inference: # inference config
    - parameters:
        - LAI
        - soil_moisture
    - optical_operator_library: some_operator.nc   # Optional
    - sar_operator_library: some_other_operator.nc # Optional
```

(continues on next page)

(continued from previous page)

```
    - a: identity
    - inflation: 1e3

Prior:
# Prior section conventions

# - 1. sub-level contains all potential variables (sm, roughness, lai, ..)
#   which are asked for/being inferred from Orchestrator/Inferrence Engine
#   and for which prior information is provided.
# - 2. sub-level contains prior type (ptype). These can be commented out
#   to be omitted.

  General:
    directory_data: ./aux_data/Static/Vegetation/
  sm:
    climatology:
      dir: ./aux_data/Climatology/SoilMoisture/
    coarse:
      dir: ./aux_data/Coarse/SoilMoisture/
  clay_fraction:
    soil_map:
      file: ./aux_data/Static/SoilTexture/CLYPPT_M_sl1_250m_ll.tif
  sand_fraction:
    soil_map:
      file: ./aux_data/Static/SoilTexture/SNDPPT_M_sl1_250m_ll.tif

#   recent:
#     dir: ""
#   user1:
#     dir: "."
    # dynamic:
    #     type: dynamic
    #     model:
    #         - API
    #         - other
    # recent:
    #   aux_data = ...
    # static:
    #       type: static
  lai:
    database:
  cab:
    database:
    #climatology:
    #  database: ../aux_data/new_geotiff
    # model:
  # veg:
    # veg_pft:
    #   type: pft
    #   database: /aux_data/some_DB
    # veg_spec:
    #   type: species
    #   database: /user_data/some_DB
    # -
```

Internal Flow

The internal flow and relations can be seen in figure 4.

Fig. 7: Figure 6: Prior Engine relationships

3.5.3 References

CHAPTER

FOUR

DEVELOPER DOCUMENTATION

4.1 Changelog

All notable changes to this project will be documented in this file.

Unreleased changes

4.1.1 Version 0.5.0 - 2019-09-19

Added

- User defined vegetation priors from the TRYdatabase
- User prior generation CLI
- helper functions to manually create soil moisture prior data from SMAP and ESA CCI data

Changed

- Documentation and README update
- documentation requirements integrated in module requirements.txt file (necessary for building on RTD)
- Bugfixes

4.1.2 Version 0.4.2 - 2018-11-05

Changed

- minor fixes in README and documentation

4.1.3 Version 0.4.1 - 2018-11-02

Added

- In code documentation Vegetation Prior

Changed

- big update on general documentation
- config file is read from `package_ressources`
- prior .vrt files are now always global

4.1.4 Version 0.4 - 2018-09-01

Added

- command line interface to allow user to add prior data
- first implementation of coarse resolution soil moisture prior based on SMAP L4 data
- averaging and aggregation of output if multiple rasters are available for one date or variable
- logging in prior engine

Changed

- prior engine framework
  - sub-engine from entry points in `setup.py`
  - conventions through abstract base class implementation in prior creator
- in-code documentation
- fixed travis installation

Removed

- –

4.1.5 Version 0.3 - 2018-03-07

Added

- *get_mean_state_vector* returns path to prior files and routes to specific submodule for soil and vegetation related priors respectively to produce/provide information.
- Vegetation prior:
  - global vegetation trait maps as static prior implemented
- Soil moisture prior:
  - basic implementation of ESA CCI soil moisture climatology based prior

Changed

- –

Removed

- –

The format is based on ‘Keep a Changelog <http://keepachangelog.com/en/1.0.0/>‘_ and this project adheres to ‘Semantic Versioning <http://semver.org/spec/v2.0.0.html>‘_.

4.2 How to contribute

You are very welcome to contribute to the MULTIPLY prior engine and we would love to see your ideas. Wether you want to make changes, allowing the engine to work in your environment or to extend the functionality of it, it should be straightforward and as easy as possible. The few guidelines which need to be followed by the contributor are listed below. To keep it simple we follow the ‘standard procedure’ on github.

4.2.1 Introduction to `git` and `Github`

Resources for learning git:

- Git Handbook
- Understanding the GitHub Flow
- Further Git and GitHub learning resources

4.2.2 Getting Started

- Make sure you have a GitHub account.
- Fork (guide) the repository on GitHub (and `git clone` your fork locally).
- Check out the repository.
- Read How to submit a contribution.

4.2.3 General Information on pull requests

from https://opensource.guide:

> You should usually open a pull request in the following situations:
>
> - Submit trivial fixes (for example, a typo, a broken link or an obvious error)
> - Start work on a contribution that was already asked for, or that you’ve already discussed, in an issue

Tips and guidelines:

- sync your fork (guide) often with the upstream repository to avoid merge conflicts.
- adhere to the GitHub Flow and create a meaningful branch for your changes
- reference relevant issues in your pull request (e.g. ‘Closes #21.’)

4.2.4 Contributing to Issues

You can contribute either by helping to solve existing issues and provide the code updates via pull request or by filing new issues.

from https://opensource.guide:

> You should usually open an issue in the following situations:
>
> - Report an error you can't solve yourself
> - Discuss a high-level topic or idea (for example, community, vision or policies)
> - Propose a new feature or other project idea

In any case:

- Check if the issue you are going to file already exists in our open issues .
- If you can't find your issue already, open a new one.

4.2.5 Contributing to Code

New features and bug fixes are very welcome. But, pull requests can only be accepted if:

- all continuous integration builds pass and
- tests for new code sections are included.

4.2.6 Contributing to Documentation

Contributions to the documentation of the MULTIPLY prior engine are always welcome. The current version can be found at http://multiply.readthedocs.io/.

After forking the repo, please find the documentation files inside the `/doc` folder in the root path of the repository. Adjust and file a pull request like you would do with code updates.

4.3 Testing

We use *PyTest* in the MULTIPLY software. The test files are located in the *test* folder in the source directory.

They can be run e.g. via:

```
pytest -vs
```

Note: This section will describe testing routines used in the prior engine necessary for development.

4.4 Module documentation

4.4.1 prior_engine module

Prior Engine for MULTIPLY.

Copyright (C) 2019 Thomas Ramsauer

class multiply_prior_engine.prior_engine.**PriorEngine**(***kwargs*)
Bases: object

Prior Engine for MULTIPLY.

holds prior initialization methods (e.g. config loading). calls specific submodules (soilmoisture_prior, vegetation_prior, ..)

_check()
initial check for passed values of - config - datestr - variables

Returns

-

Return type

-

_concat_priors(*prior_dict*)
Concatenate individual state vectors and covariance matrices for sm, veg, ..

Returns dictionary with keys beeing superordinate prior name (sm, ..)

Return type dictionary

_get_prior(*var*)
Called by get_priors for all variables to be inferred. For specific variable/prior (e.g. sm climatology) get prior info and calculate/provide prior.

Parameters **var** – prior name (e.g. sm, lai, ..)

Returns

-

Return type

-

default_config = '/home/docs/checkouts/readthedocs.org/user_builds/multiply-prior-engi

get_priors()
Get prior data. calls *_get_prior* for all variables (e.g. sm, lai, ..) passed on to get_mean_state_vector method.

Returns dictionary with prior names/prior types/filenames as {key/{key/values}}.

Return type dictionary of dictionary

multiply_prior_engine.prior_engine.**_get_config**(*configfile*)
Load config from self.configfile. writes to self.config.

Returns

-

multiply_prior_engine.prior_engine.**get_mean_state_vector**(*datestr: str, variables: list, config: str = './sample_config_prior.yml'*) → dict
Return dictionary with variable dependent sub dictionary with prior type (key) and filenames of prior files (values).

Parameters

- **datestr** – The date (time?) for which the prior needs to be derived

- **variables** – A list of variables (sm, lai, roughness, ..)

for which priors need to be available

Returns dictionary with keys being the variables and

values being a dictionary of prior type and filename of prior file.

4.4.2 prior module

Prior Class for MULTIPLY.

Copyright (C) 2018 Thomas Ramsauer

class multiply_prior_engine.prior_creator.**PriorCreator**(***kwargs*)
Bases: object

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 211

_abc_registry = <_weakrefset.WeakSet object>

_check()

_create_datetime()

_create_time_vector()
Creates a time vector dependent on start & end time and time interval from config file. A vector containing datetime objects is written to self.time_vector. A vector containing months ids (1-12) for each timestep is written to self.time_vector_months.

Returns

-

Return type

-

abstract compute_prior_file() → str
Might perform some computation, then retrieves the path to a file containing the prior info :return:

abstract classmethod get_variable_names() → List[str]

Returns A list of the variables that this prior creator is able to create priors for

4.4.3 soilmoisture_prior module

Soil Priors for Prior Engine in MULTIPLY.

Copyright (C) 2019 Thomas Ramsauer

class multiply_prior_engine.soilmoisture_prior_creator.**MapPriorCreator**(***kwargs*)
Bases: *multiply_prior_engine.prior_creator.PriorCreator*

Not Implemented Prior which is based on a LC map and a LUT

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 211

_abc_registry = <_weakrefset.WeakSet object>

classmethod get_variable_names()

Returns A list of the variables that this prior creator is able to create priors for

class multiply_prior_engine.soilmoisture_prior_creator.**RoughnessPriorCreator**(***kwargs*)

Bases: *multiply_prior_engine.soilmoisture_prior_creator.MapPriorCreator*

Not Implemented Roughness Prior Creator which is based on a LC map and a LUT

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 211

_abc_registry = <_weakrefset.WeakSet object>

_map_lut()

should do the mapping of s, l, ACL type

_read_lc()

_read_lut()

calc()

compute_prior_file()

Might perform some computation, then retrieves the path to a file containing the prior info :return:

classmethod get_variable_names()

Returns A list of the variables that this prior creator is able to create priors for

save()

save mapped roughness data to file

class multiply_prior_engine.soilmoisture_prior_creator.**SoilMoisturePriorCreator**(***kwargs*)

Bases: *multiply_prior_engine.prior_creator.PriorCreator*

Soil moisture prior class. Calculation of climatological prior.

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 211

_abc_registry = <_weakrefset.WeakSet object>

_calc_climatological_prior()

Calculate climatological prior. Reads climatological file and extracts proper values for given timespan and -interval. Then converts the means and stds to state vector and covariance matrices.

Returns state vector and covariance matrix

Return type tuple

_check_gdal_compliance(*fn*)

_create_global_vrt(*fn*, *local=True*)

Create VRT file for file.

By default, the .vrt-file will be written to a local temporary directory. If *local* is set to False, the file is written to the directory the input file (fn) currently lives in.

Parameters

- **`fn`** – file name
- **`local`** – create temporary local vrt.

Returns file name of created vrt, or initial file name if no success.

Return type string

`_extract_climatology`()
Extract climatology values for ROI. Part of _clac_climatological_prior().

`_get_climatology_file`()
Load pre-processed climatology into self.clim_data. Part of prior._calc_climatological_prior().

`_get_prior_file_from_dir`(*directory*, *return_vrt=True*)
Get filename(s) of prior file(s) from directory. If multiple files are found self._merge_multiple_prior_files is called.

Currently, the following prior types are supported: - climatology (calculated from ESA CCI data, standard) - coarse (daily aggregated SMAP L4 data, standard) - soil_map (gloabal soil texture map data from soilgrids.org) - user prior, provided through user_prior_creator

Parameters **`directory`** – directory containing the files (from config)

Returns filename

Return type string

`_get_recent_sm_proxy`()

`_merge_multiple_prior_files`(*fn_list*)
Merge files if more than one is available for current time step. should be obsolete.

Parameters **`fn_list`** – file list to process

Returns file name of merged file

Return type string

`_provide_prior_file`()
Provide variable and prior type specific prior file name to Prior Engine.

Returns absolute path to prior file for requested prior.

The file is gdal-compatible to be used in inference engine - either GeoTiff or VRT format. It includes 2 bands:

1. mean value raster
2. uncertainty raster

Return type string

`compute_prior_file`()
Initialize prior specific (climatological, . . .) calculation.

Returns filename of prior file

Return type string

`classmethod get_variable_names`()

Returns A list of the variables that this prior creator is able to create priors for

4.4.4 vegetation_prior module

class multiply_prior_engine.vegetation_prior_creator.**VegetationPriorCreator**(***kwargs*)

Bases: *multiply_prior_engine.prior_creator.PriorCreator*

Description

AssignPFTTraits2Map(*PFT*, *PFT_ids*, *varnames*)

Create Vegetation trait Prior map, using the Trait-database and PFT distribution maps This function sets up a parallel processing chain around - processespercore: here the actual assignment of traits to PFT distributions is performed

Parameters

- **PFT** – arrays containing global Maps of PFT distributions
- **PFT_ids** – a list containing PFT ids
- **varnames** – list of variables to be converted into global file

Returns map of vegetation trait-averages per PFT id, map of vegetation trait-uncertainties per PFT id

Return type

Combine2PFT(*LCC_map*, *CLM_map_i*)

Create PFT maps using CCI Landcover and Koppen Climate zone information :param LCC_map: CCI Landcover map :param CLM_map_i: Regridded Koppen Climate Zone map :returns: PFT occurrence map, PFT classes, Number of PFTs, PFT ids :rtype:

CombineTiles2Virtualfile(*variable*, *doystr*, *directory_data*)

Combine all geotiff files into a virtual global file

Parameters

- **variable** – variable to be converted into global file
- **doystr** – string containing date&time '2007-12-31 04:23' for which global file needs to be created

Returns the filename of the global VRT file

Return type

CreateDummyDatabase()

create netcdf Database files to hold database values

Returns

-

Return type

CreateRealDatabase()

DownloadCrossWalkingTable()

Download Crosswalking table Here the Cross walking table is downloaded to create the CCI landcover map. At the moment this is simply a placeholder for future functionality.

Returns

-

Return type

DynamicProcessing(*varnames*, *LCC_lon*, *LCC_lat*, *Prior_pbm_avg*, *Prior_pbm_unc*, *doystr*, *write_output=True*)

Extending Peak Biomass (PBM) traits to seasonal Priors At this moment, this function is only a placeholder for the later implementations. The final implementation will be modelled using - covariances between traits and (seasonal) meteorological variables - phenological evolution (trained using plant growth models)

Parameters

- **varnames** – list of variables to be converted into global file
- **LCC_lon** – array with longitude values of (subsetted tile of) study area
- **LCC_lat** – array with latitude values of (subsetted tile of) study area
- **Prior_pbm_avg** – Vegetation Traits mean value at PBM
- **doystr** – string containing date&time '2007-12-31 04:23' for processing needs to be performed
- **Prior_pbm_unc** – Vegetation Traits uncertainty value at PBM
- **write_output** – Binary Value (TRUE/FALSE) controlling the writing of outputfiles

Returns

-

Return type

ExtractPFT4TryDatabaseEntries(*Lat_*, *Lon_*, *Plantgroup_*, *Crop_*, *LeafType_*, *C3C4_*, *LeafPhen_*)

OfflineProcessing()

Creation of LCC landcover map This :returns: :rtype:

PhenologicalEvolution(*Prior_pbm_avg*, *Prior_pbm_unc*, *doystr*, *Meteo_map_i=None*)

Model the Phenological Evolution of Vegetation traits This function is a placeholder to be used when the dynamic functionality is created.

Parameters

- **Prior_pbm_avg** – Vegetation trait-averages at Peak Biomass
- **Prior_pbm_unc** – Vegetation trait-uncertainty (@PBM)
- **doystr** – string containing date&time '2007-12-31 04:23' for processing needs to be performed
- **Meteo_map_i** – Place_holder for meteorological data-files

Returns Temporal Prior-averages, Temporal Prior-uncertainties

Return type

ProcessData(*variables=None*, *state_mask=None*, *timestr='2005-05-05 05:55'*, *logger=None*, *file_prior=None*, *file_lcc=None*, *file_biome=None*, *file_meteo=None*)

Process Data Apriori Calculation of prior using Databases of Vegetation Traits. This function is split into two parts (which are run for all Tiles over the study are) - OfflineProcessing: This only has to be performed once (to make sure all the input data is available) - StaticProcessing: Creating Peak Biomass (PBM) Priors - DynamicProcessing: Extending PBM traits to seasonal priors (a placeholder for the later implementations)

Parameters

- **variables** – list of variables to be converted into global file
- **state_mask** – place holder for spatial mask (not implemented)

- **timestr** – string containing date&time '2007-12-31 04:23' for which global file needs to be created
- **logger** – log-file for capturing message from the scripts
- **file_prior** – place-holder for prior (TRY) database - filename (at the moment hardcoded)
- **file_lcc** – place-holder for landcover data - filename (at the moment hardcoded)
- **file_biome** – place-holder for biome data - filename (at the moment hardcoded)
- **file_meteo** – place-holder for meteorological data - filename (at the moment hardcoded)

Returns filenames to global VRT prior files

Return type

ReadClimate()

Read Climate Zone information A Climate Zone map (created on basis of the Koppen Climatic Zone classification)is read.

Returns climate zone map, longitude, latitude, climate zone classes

Return type

ReadLCC()

Read Landcover information The Landcover map from the Climate Change Initiaive (CCI) is read.

Returns landcover map, longitude, latitude, landcover class names

Return type

ReadMeteorologicalData(*doystr*)

Read Meteorological Variables This function is a placeholder to be used when the dynamic functionality is created.

Parameters **doystr** – string containing date&time '2007-12-31 04:23' for processing needs to be performed

Returns Meteorological data (to be used for upscaling Peak Biomass traits to seasonal priors)

Return type

ReadTraitDatabase(*varnames*, *pft_id=1*)

Read Traits from Database A local (modified) version of the Try Database (containing vegetation traits) is read.

Parameters

- **varnames** – list of variables to be converted into global file
- **pft_id** – list of pft id numbers for which the traits needs to be read.

Returns an array of Traits per PFT group

Returns an array of Traits per PFT group

Return type

ReadTryDatabase()

ReadTryFile()

RescaleCLM(*CLM_lon*, *CLM_lat*, *CLM_map*, *LCC_lon*, *LCC_lat*)
Collocate Climate Zone map with landcover coordinates The Climate Zone map has a different resolution/grid than the Landcover map. This preprocessing is performed to collocate both (in order to facilitate the merging downstream.)

Parameters

- **CLM_lon** – array containing the longitude values of the Climate Zone map
- **CLM_lat** – array containing the latitude values of the Climate Zone map
- **CLM_map** – array containing the Climate Zone map
- **LCC_lon** – array containing the longitude values of the CCI Landcover map
- **LCC_lat** – array containing the latitude values of the CCI Landcover map

Returns array containing the Regridded Climate Zone map

Return type

RunCrossWalkingTable(*Path2CWT_tool=None*, *Path2LC=None*)
Creating CCI landcover maps (using crosswalking table).

please note that to run the crosswalking tool, the specific requirements for BEAM need to be met (java64bit + . . .)

Parameters

- **Path2CWT_tool** –
- **Path2LC** –

Returns

-

Return type

StaticProcessing(*varnames*, *write_output=False*)
Creating Peak Biomass (PBM) Priors Priors are created by upscaling vegetation traits obtained through the TRY database. Within the TRY database vegetation traits are provided per PFT group. In order to upscale these values, a global PFT map is required. This is created by merging a global Landcover map (from Climate Change Initiative, CCI) with a climate zone map (using the Koppen classification). This is accomplished by -ReadLCC: Reading the CCI Landcover map -ReadClimate: Reading the Koppen Climate zone map -RescaleCLM: Rescaling Climate zone map to collocate with Landcover CCI. -Combine2PFT: Combining Climate zone + Landcover maps into PFTs Using this global PFT map, the values from the TRY database are afterwards spatially distributed by -AssignPFTTraits2Map: assigning and aggregating traits to PFT maps.

Parameters

- **varnames** – list of variables to be converted into global file
- **write_output** – Binary value (TRUE/FALSE) controlling the writing of outputfiles

Returns longitude, latitude, Prior_avg, Prior_unc

Return type

WriteGeoTiff(*LCC_lon*, *LCC_lat*, *Prior_avg*, *Prior_unc*, *doystr='static'*)
Write Vegetation Prior data (mean/unc) to GEOTIFF outputfiles. :param LCC_lon: longitude of the Prior data (same as used Landcover map) :param LCC_lat: latitude of the Prior data (same as used Landcover map) :param Prior_avg: Vegetation prior average values :param Prior_unc: Vegetation prior uncertainty values :param doystr: string containing date&time '2007-12-31 04:23' for data to be written :returns: -

WriteOutput (*LCC_lon*, *LCC_lat*, *Prior_avg*, *Prior_unc*, *doystr='static'*)
Write Vegetation Prior data (mean/unc) to NETCDF outputfiles. This functionality is obsolete as all outputs are written to GeoTiff files

Parameters

- **LCC_lon** – longitude of the Prior data (same as used Landcover map)
- **LCC_lat** – latitude of the Prior data (same as used Landcover map)
- **Prior_avg** – Vegetation prior average values
- **Prior_unc** – Vegetation prior uncertainty values
- **doystr** – string containing date&time '2007-12-31 04:23' for data to be written

Returns

-

Return type

_abc_cache = <_weakrefset.WeakSet object>

_abc_negative_cache = <_weakrefset.WeakSet object>

_abc_negative_cache_version = 212

_abc_registry = <_weakrefset.WeakSet object>

compute_prior_file()
Combine Tiles into single Prior VRT file

Returns filename of specific VRT file

Return type

classmethod get_variable_names()

Returns A list of the variables that this prior creator is able to create priors for

multiply_prior_engine.vegetation_prior_creator.**_get_config**(*configfile*)
Load config from self.configfile. writes to self.config.

Returns

-

multiply_prior_engine.vegetation_prior_creator.**fun**(*f*, *q_in*, *q_out*)

multiply_prior_engine.vegetation_prior_creator.**parmap**(*f*, *X*, *nprocs=4*)
Enable Parallel processing This code is created to enable parallel processing with python

Parameters

- **f** – function to be called
- **X** – input to the function
- **nprocs** – number of cores to be used

Returns output of function

Return type

multiply_prior_engine.vegetation_prior_creator.**processespercore**(*varname*, *PFT*, *PFT_ids*, *VegetationPriorCreator*)

Create Prior values from PFT distributions and Vegetation traits For each PFT the specific trait (according to varname) are read from the Trait-Database. These traits are then statistically analysed to produce the mean and standard deviations. These trait values are then evaluated against the PFT distribution (occurrence) map and joint together to create a single Prior (mean&uncertainty) estimate for each spatial location

Please note that: This function is encapsulated within the parmap method to run in parallel on different cores

Parameters

- **varname** – variable to be processed
- **PFT** – arrays containing global Maps of PFT distributions
- **PFT_ids** – a list containing PFT ids
- **VegetationPriorCreator** – class containing all the functionality to be run (per core)

Returns Vegetation Prior average values, Vegetation Prior uncertainty values

Return type

4.5 License


```
                              Preamble


The GNU General Public License is a free, copyleft license for
```

software and other kinds of works.

The licenses for most software and other practical works are designed

to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not

price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you

these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps:

(1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains

that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run

modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents.

States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and

modification follow.

```
TERMS AND CONDITIONS
```

1. Definitions.

   "This License" refers to version 3 of the GNU General Public License.

   "Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

   "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

   To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

   A "covered work" means either the unmodified Program or a work based on the Program.

   To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

   To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

   An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

2. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

   The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

   The Corresponding Source for a work in source code form is that same work.

3. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

4. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

5. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

6. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

   a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   b) The work must carry prominent notices stating that it is released under this License and any conditions added under section

   1. This requirement modifies the requirement in section 4 to "keep intact all notices".

   c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

   d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

   A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

7. Conveying Non-Source Forms.

   You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

   a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

   b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

   c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

   d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

8. Additional Terms.

   "Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

   When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

9. Termination.

   You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

10. Acceptance Not Required for Having Copies.

    You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

11. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

12. Patents.

    A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

    A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

    Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

    In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

    If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

    If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

    A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

13. No Surrender of Others' Freedom.

    If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

14. Use with the GNU Affero General Public License.

    Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

15. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

16. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

17. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF

THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

18. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

```
         END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs
```

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C)

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see http://www.gnu.org/licenses/.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short

notice like this when it starts in an interactive mode:

```
<program>  Copyright (C) <year>  <name of author>
This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and`show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school,

if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see http://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program

into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read http://www.gnu.org/philosophy/why-not-lgpl.html.

CHAPTER

FIVE

INDICES AND TABLES

- modindex
- genindex
- search

BIBLIOGRAPHY

[Mattia] Mattia, F. et al. (2006) Using a priori information to improve soil moisture retrieval from ENVISAT ASAR AP data in semiarid regions. IEEE Trans. Geosci. Remote Sens. 44: 900–912.

[Dorigo] Dorigo, W. A., et al., 2017, ESA CCI Soil Moisture for improved Earth system understanding: State-of-the art and future directions, Remote Sensing of Environment, 203, 185-215, 2017, doi:10.1016/j.rse.2017.07.001.

[Gruber] Gruber, A., et al., 2017, Triple Collocation-Based Merging of Satellite Soil Moisture Retrievals, Transactions on Geoscience and Remote Sensing, 55(12), 1-13. doi:10.1109/TGRS.2017.2734070.

[Liu] Liu, Y. Y., et al., 2012, Trend-preserving blending of passive and active microwave soil moisture retrievals, Remote Sensing of Environment, 123, 280-297.

[Reichle] Reichle, R. et al. 2014. SMAP Algorithm Theoretical Basis Document: L4 Surface and Root-Zone Soil Moisture Product. SMAP Project, JPL D-66483, Jet Propulsion Laboratory, Pasadena, CA, USA.

[GRUBER2019] Gruber, A., Scanlon, T., van der Schalie, R., Wagner, W., and Dorigo, W.: Evolution of the ESA CCI Soil Moisture climate data records and their underlying merging methodology, Earth Syst. Sci. Data, 11, 717-739, https://doi.org/10.5194/essd-11-717-2019, 2019

[JPL] https://smap.jpl.nasa.gov/data/

[Mattia] Mattia, F. et al. (2006) Using a priori information to improve soil moisture retrieval from ENVISAT ASAR AP data in semiarid regions. IEEE Trans. Geosci. Remote Sens. 44: 900–912.

[Dorigo] Dorigo, W. A., et al., 2017, ESA CCI Soil Moisture for improved Earth system understanding: State-of-the art and future directions, Remote Sensing of Environment, 203, 185-215, 2017, doi:10.1016/j.rse.2017.07.001.

[Gruber] Gruber, A., et al., 2017, Triple Collocation-Based Merging of Satellite Soil Moisture Retrievals, Transactions on Geoscience and Remote Sensing, 55(12), 1-13. doi:10.1109/TGRS.2017.2734070.

[Liu] Liu, Y. Y., et al., 2012, Trend-preserving blending of passive and active microwave soil moisture retrievals, Remote Sensing of Environment, 123, 280-297.

[Reichle] Reichle, R. et al. 2014. SMAP Algorithm Theoretical Basis Document: L4 Surface and Root-Zone Soil Moisture Product. SMAP Project, JPL D-66483, Jet Propulsion Laboratory, Pasadena, CA, USA.

PYTHON MODULE INDEX

m

INDEX

Symbols

O

P

R

S

V

W