# flagit Documentation

***Release unknown***

**TU Wien**

**Nov 09, 2022**

# Contents


**1 Citation** **3**

**2 Installation** **5**

**3 Description** **7**

**4 Contribute** **9**
4.1 Guidelines . . . . . 9

**5 Note** **11**

**6 Run ISMN flag procedures** **13**

**7 Contents** **15**
7.1 License . . . . . 15
7.2 Contributors . . . . . 15
7.3 Changelog . . . . . 16
7.4 flagit . . . . . 16

**8 Indices and tables** **17**

**Python Module Index** **19**

**Index** **21**

ISMN quality control procedures for in situ soil moisture time series

# CHAPTER 1

# Citation

If you use the software in a publication then please cite:

- Dorigo, W.A. , Xaver, A. Vreugdenhil, M. Gruber, A., Hegyiova, A. Sanchis-Dufau, A.D., Zamojski, D. , Cordes, C., Wagner, W., and Drusch, M. (2013). Global Automated Quality Control of In situ Soil Moisture data from the International Soil Moisture Network. Vadose Zone Journal, 12, 3, doi:10.2136/vzj2012.0097
- https://github.com/TUW-GEO/flagit

CHAPTER 2

# Installation

For installation we recommend Miniconda. So please install it according to the official instructions. As soon as the `conda` command is available in your shell you can continue:

```
conda install -c conda-forge pandas scipy numpy
```

This following command will install the flagit pip package:

```
pip install flagit
```

To create a full development environment with conda, the environment.yml file in this repository can be used:

```
git clone git@github.com:TUW-GEO/flagit.git flagit
cd flagit
conda create -n flagit python=3.7 # or any supported python version
conda activate flagit
conda env update -f environment.yml -n flagit
python setup.py develop
```

After that you should be able to run:

```
python setup.py test
```

to run the test suite.

# CHAPTER 3

## Description

The International Soil Moisture Network (ISMN) quality control procedures are used to detect implausible and dubious measurements in hourly situ soil moisture time series. When downloading data at ISMN all variable-data are provided with additional tags in column “qflag”, which can be one of three main categories: C (exceeding plausible geophysical range), D (questionable/dubious) or G (good).

| code | description | ancillary data required |
|---|---|---|
| C01 | soil moisture < 0 $m^3/m^3$ | |
| C02 | soil moisture > 0.60 $m^3/m^3$ | |
| C03 | soil moisture > saturation point (based on HWSD) | HWSD sand, clay and organic content |
| D01 | negative soil temperature (in situ) | in situ soil temperature |
| D02 | negative air temperature (in situ) | in situ air temperature |
| D03 | negative soil temperature (GLDAS) | GLDAS soil temperature |
| D04 | rise in soil moisture without precipitation (in situ) | in situ precipitation |
| D05 | rise in soil moisture without precipitation (GLDAS) | GLDAS precipitation |
| D06 | spikes | |
| D07 | negative breaks (drops) | |
| D08 | positive breaks (jumps) | |
| D09 | constant low values following negative break | |
| D10 | saturated plateaus | |
| G | good | |

At ISMN, ancillary data sets are used for flags C03, D01 - D05 (see table above). Since we do not provide ancillary data, we kindly ask users to either provide their own ancillary in situ and GLDAS data (including a soil moisture saturation value for flag C03) in the input (pandas.DataFrame), or accept the limitation of the quality control to flags without ancillary requirements.

We hope to update the functionality of this package to facilitate the inclusion of ancillary data.

For a detailed description of the quality control procedures see paper on Global Automated quality control.

CHAPTER 4

# Contribute

We would be happy if you would like 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 ismn repository to your account
- Clone the repository
- make a new feature branch from the ismn master branch
- Add your feature
- Please include tests for your contributions in one of the test directories. We use unittest so a simple function called test_my_feature is enough
- submit a pull request to our master branch

# CHAPTER 5

## Note

This project has been set up using PyScaffold 3.2.3. For details and usage information on PyScaffold see https://pyscaffold.org/.

# CHAPTER 6

## Run ISMN flag procedures

This example program shows how to initialize the Interface and run the flagging procedures.

As Input a pandas.DataFrame of the following format is required:

| | soil_moisture | soil_temperature | air_temperature | precipitation | gldas_soil_temperature | gldas_precipitation |
|---|---|---|---|---|---|---|
| utc | | | | | | |
| 2017-01-27 00:00:00 | 5.0 | -4.7 | -13.6 | 0.0 | -8.4 | 0.0 |
| 2017-01-27 01:00:00 | 4.9 | -4.9 | -13.4 | 0.0 | -8.6 | 0.0 |
| 2017-01-27 02:00:00 | 4.9 | -5.1 | -14.0 | 0.0 | -8.8 | 0.0 |
| 2017-01-27 03:00:00 | 4.9 | -5.1 | -13.2 | 0.0 | -8.9 | 0.0 |
| 2017-01-27 04:00:00 | 4.9 | -4.9 | -11.2 | 0.0 | -9.1 | 0.0 |
| 2017-01-27 05:00:00 | 4.9 | -4.6 | -10.1 | 0.0 | -9.2 | 0.0 |
| 2017-01-27 06:00:00 | 5.0 | -4.5 | -8.9 | 0.0 | -9.4 | 0.0 |

```
from flagit import flagit
import pandas as pd
```

```
# read from CSV file
file_path = '/path_to_dataframe/*.csv'
df = pd.read_csv(file_path, index_col='utc', parse_dates=True)
```

```
# initialize interface and run all flagging procedures
flag = flagit.Interface(df)
```

(continues on next page)

(continued from previous page)

```
result_df = flag.run(sat_point = 42.7)

# alternatively: choose only specific procedures by providing a list or string as␣
↪name:
flag = flagit.Interface(df)
result_df = flag.run(name = ['D06', 'D07', 'D09'])
result_df = flag.run(name = 'C01')
```

```
# get flag-descriptions
flag = flagit.Interface(df)
flag.get_flag_description()
```

CHAPTER 7

# Contents

## 7.1 License



## 7.2 Contributors

- Daniel Aberer <daniel.aberer@geo.tuwien.ac.at>
- Angelika Xaver <angelika.xaver@geo.tuwien.ac.at>
- Wolfgang Preimesberger <wolfgang.preimesberger@geo.tuwien.ac.at>

## 7.3 Changelog

### 7.3.1 Version 0.1

- First public version of package

### 7.3.2 Version 0.2

- Adapt code for internal use
- Bugfix D06 rolling mean calculation
- Bugfix case of multiple precipitation sensors

## 7.4 flagit

### 7.4.1 flagit package

#### Submodules

#### flagit.flagit module

#### flagit.settings module

**class** flagit.settings.**Variables**
Bases: object

Class for flagging thresholds of soil moisture, accompanying ancillary observations and

**ancillary_p_min = 0.2**

**ancillary_ta_lower = 0**

**ancillary_ts_lower = 0**

**hi_boundary**(*var*)
Upper threshold for quality control units: soil moisture : $m^3/m^3$ soil temperature, air temperture, surface temperature : degree Celsius precipitation, snow water equivalent, snow depth : mm soil suction : kPa

var : string variable name (some examples are: soil_moisture, soil_temperature, snow_water_equivalent)

**low_boundary**(*var*)
Lower threshold for quality control units: soil moisture : $m^3/m^3$ soil temperature, air temperture, surface temperature : degree Celsius precipitation, snow water equivalent, snow depth : mm soil suction : kPa

var : string variable name (some examples are: soil_moisture, soil_temperature, snow_water_equivalent)

**plateau_count = 0**

#### Module contents

CHAPTER 8

# Indices and tables

- genindex
- modindex
- search

# Python Module Index

## f

flagit, 16
flagit.settings, 16

# Index


## A

ancillary_p_min (*flagit.settings.Variables attribute*), 16
ancillary_ta_lower (*flagit.settings.Variables attribute*), 16
ancillary_ts_lower (*flagit.settings.Variables attribute*), 16

## F

flagit (*module*), 16
flagit.settings (*module*), 16

## H

hi_boundary() (*flagit.settings.Variables method*), 16

## L

low_boundary() (*flagit.settings.Variables method*), 16

## P

plateau_count (*flagit.settings.Variables attribute*), 16

## V

Variables (*class in flagit.settings*), 16