evaporation Documentation

Release 1.0.1

Antonis Christofides

Sep 01, 2021

Contents:

CHAPTER 1

Usage

1.1 Synopsis

```
vaporize [--traceback] config_file
```

1.2 Description and quick start

`vaporize` calculates evapotranspiration with the Penman-Monteith method. It works either with GeoTIFF files or with time series files. In either case, it reads files with temperature, humidity, solar radiation, pressure and wind speed, and produces a file with evapotranspiration. The details of its operation are specified in the configuration file specified on the command line.

The methodology used is that of Allen et al. (1998). Details can be found in *API* and in the code itself, which has comments indicating which equations it uses.

1.2.1 Installation

```
pip install evaporation
```

1.2.2 How to run it

First, you need to create a configuration file with a text editor such as `vim`, `emacs`, `notepad`, or whatever. Create such a file and name it, for example, `/var/tmp/vaporize.conf`, with the following contents (the contents don't matter at this stage, just copy and paste them from below):

```
loglevel = INFO
```

Then, open a command prompt and give it this command:

```
vaporize /var/tmp/vaporize.conf
```

If you have done everything correctly, it should output an error message complaining that something in its configuration file isn't right.

1.2.3 Configuration file example

Take a look at the following example configuration file and read the explanatory comments that follow it:

```
loglevel = INFO
logfile = C:\Somewhere\vaporize.log
base_dir = C:\Somewhere
albedo = 0.23
nighttime_solar_radiation_ratio = 0.8
elevation = 8
time_step = H
unit_converter_pressure = x / 10.0
unit_converter_solar_radiation = x * 3600 / 1e6
```

With the above configuration file, `vaporize` will log information in the file specified by *`logfile`*. It will calculate hourly evaporation (*`time_step`*) at the specified *`elevation`* with the specified *`albedo`* and *`nighttime_solar_radiation_ratio`* (these three parameters can be GeoTIFF files instead of numbers). For some variables, the input files are in different units than the default ones (hPa instead of kPa for pressure, W/m^2 instead of MJ/m^2/h for solar radiation) and need to be converted (*`unit_converter`*).

If the *`base_dir`* contains `tif` files, the calculation is performed once for each one of the sets of files; for example, if inside *`base_dir`* there are files `temperature-2014-10-12-18-00+0200.tif`, `humidity-2014-10-12-18-00+0200.tif`, and so on (including variables named `wind_speed`, `pressure`, and `solar_radiation`), there will be a resulting file `evaporation-2014-10-12-18-00+0200.tif`; if there are files for other dates, there will be a result for them as well. The calculation is performed only if the resulting file does not already exist, or if at least one of the input files has a later modification time. If there are any `evaporation-....tif` files without corresponding input files, they will be deleted.

If the *`base_dir`* contains `hts` files, the calculation is performed for these time series. For example, if inside *`base_dir`* there are files `temperature.hts`, `humidity.hts`, and so on, there will be a resulting file `evaporation.hts`, overwriting any previously existing such file.

1.3 Configuration file reference

The configuration file has the format of INI files, but without sections.

1.3.1 Parameters

loglevel
Optional. Can have the values `ERROR`, `WARNING`, `INFO`, `DEBUG`. The default is `WARNING`.

logfile
Optional. The full pathname of a log file. If unspecified, log messages will go to the standard error.

base_dir
The directory in which `vaporize` will look for input files and write output files. If unspecified, it is the directory from which `vaporize` was started.

time_step
A string specifying the time step. In this version, `vaporize` can only handle hourly (H) or daily (D) time steps.

elevation
Meters of the location above sea level; this can be either a number or a GeoTIFF file with a digital elevation model.

nighttime_solar_radiation_ratio
(Hourly step only.)

In order to estimate the outgoing radiation, the ratio of incoming solar radiation to clear sky solar radiation is used as a representation of cloud cover. This, however, does not work during the night, in which case *nighttime_solar_radiation_ratio* is used as a rough approximation of that ratio. It should be a number between 0.4 and 0.8; see Allen et al. (1998), top of page 75. It can be a number or a GeoTIFF file.

albedo
A number between 0 and 1 or a GeoTIFF file with such numbers. It can also be a list of twelve space-separated numbers and/or GeoTIFF files, where the first is for January, the second for February, and so on. For example:

```
albedo = albedo-jan.tif albedo-feb.tif albedo-mar.tif albedo-apr.tif
         albedo-may.tif albedo-jun.tif albedo-jul.tif albedo-aug.tif
         albedo-sep.tif 0.23           albedo-nov.tif albedo-dec.tif
```

Note that in the configuration file long lines can be wrapped by indenting the additional lines. Also note that GeoTIFF files can be mixed with numbers; in the above example, GeoTIFF files are specified for all months except for October, which has a single value of 0.23.

If a single number or GeoTIFF file is specified, it is used for all the year.

unit_converter
The meteorological values that are supplied with the input files of the file set sections are supposed to be in the following units:

Parameter	Unit
temperature	°C
humidity	%
wind speed	m/s
pressure	kPa
solar radiation	$MJ/m^2/step$
sunshine duration	h

If they are in different units, `unit_converter_temperature`, `unit_converter_humidity`, and so on, are Python expressions that convert the given units to the above units; in these expressions, the symbol `x` refers to the given value. For example, if you have temperature in , specify:

```
unit_converter_temperature = (x - 32.0) * 5.0 / 9.0
```

Use 32.0 rather than 32, and so on, in order to ensure that the calculations will be performed in floating point.

You can also use this to convert wind speed to a different height. Wind speed at 2 m from the ground is required. If you have wind speed at a different height, convert it using Eq. 47, p. 56, of Allen et al. (1998). For example, if you have wind speed at 10 m, specify this:

> unit_converter_wind_speed = x * 4.87 / math.log(67.8 * 10 - 5.42)

temperature_prefix
temperature_max_prefix

temperature_min_prefix
humidity_prefix
humidity_max_prefix
humidity_min_prefix
wind_speed_prefix
pressure_prefix
solar_radiation_prefix
sunshine_duration_prefix
evaporation_prefix

Optional. `vaporize` assumes that the input files are named `*variable-date*.tif` or `*variable*.hts`, where *variable* one of `temperature`, `temperature_max`, `temperature_min`, `humidity`, `humidity_max`, `humidity_min`, `wind_speed`, `pressure`, `solar_radiation`, and `sunshine_duration`, and, similarly, for the output file *variable* is `evaporation`. With these parameters these names can be changed; for example:

```
humidity_prefix = hum
```

In that case, the humidity files are going to have a name similar to `hum-2014-10-12-18-00+0200.tif` (for hourly) or `hum-2014-10-12.tif` (for daily).

`vaporize` will use the pressure if it is available in the input files, otherwise it will calculate it from the elevation.

1.4 References

R. G. Allen, L. S. Pereira, D. Raes, and M. Smith, Crop evapotranspiration - Guidelines for computing crop water requirements, FAO Irrigation and drainage paper no. 56, 1998.

CHAPTER 2

API

```
from evaporation import PenmanMonteith
```

class PenmanMonteith(*albedo*, *elevation*, *latitude*, *time_step*, *longitude=None*, *nighttime_solar_radiation_ratio=None*, *unit_converters={}*)

Calculates evapotranspiration according to the Penman-Monteith equation. The methodology used is that of Allen et al. (1998). Details can be found in the code, which has comments indicating which equations it uses.

First the class is initialized with some parameters that are constant for the area of interest; then, the `calculate()` method can be called as many times as necessary in order to calculate reference evapotranspiration given the date and time and the values of the meteorological variables.

Evapotranspiration can be calculated either at a point or on a grid, so the input can be either simple scalar values or `numpy` arrays. The term "scalar or array", when used below, signifies a parameter that can be either. You will normally either use all scalars or all arrays; however, when you generally use arrays, you may also use scalars for some of the parameters if you want the array to have the same value for all gridpoints; for example, you might want to have a single albedo value for all gridpoints.

The class is initialized with the following parameters:

albedo is either a scalar or array, or a sequence of 12 scalars or arrays. If it is a sequence, the first item is the albedo in January, the second is for February, and so on. If it is a single scalar or array, it is used for the entire year. The albedo is a number between 0 and 1.

elevation is a scalar or array with the location elevation above sea level in meters.

latitude and *longitude* are scalars or arrays, in decimal degrees north of the equator or east of the prime meridian (negative for west or south). Only *latitude* needs to be specified for calculating daily evaporation.

time_step is a string: "D" for daily, "H" for hourly.

In order to estimate the outgoing radiation, the ratio of incoming solar radiation to clear sky solar radiation is used as a representation of cloud cover. However, when calculating hourly evaporation, this does not work during the night, in which case *nighttime_solar_radiation_ratio* is used as a rough approximation of that ratio. It should be a scalar or array between 0.4 and 0.8; see Allen et al. (1998), top of page 75.

The meteorological values that will be supplied after class initialization to the `calculate()` method are supposed to be in the following units:

Parameter	Unit
temperature	°C
humidity	%
wind speed	m/s
pressure	kPa
solar radiation	$MJ/m^2/h$

If they are in different units, *unit_converters* is a dictionary with functions to convert them. For example, if you have pressure in hPa and solar radiation in W/m^2, you should specify this:

```
unit_converters = {
    'pressure': lambda x: x / 10.0,
    'solar_radiation': lambda x: x * 3600 / 1e6,
}
```

Any variable whose name is not found in *unit_converters* is used as is, without conversion.

calculate(*self*, ***kwargs*)

Calculates and returns the reference evapotranspiration in mm.

For daily step, the keyword arguments must be *temperature_max*, *temperature_min*, *humidity_max*, *humidity_min*, *wind_speed*, *adatetime*, and one of *solar_radiation* or *sunshine_duration*. *adatetime* must be a `date` object, not a `datetime` object, but it is named *adatetime* for consistency with the hourly step. The result is the reference evapotranspiration for the given day.

For hourly step, the keyword arguments must be *temperature*, *humidity*, *wind_speed*, *solar_radiation*, *adatetime*, and, optionally, *pressure* (if the pressure is not specified it is calculated from the elevation). The result is the reference evapotranspiration for the hour that ends at *adatetime*, which must be a timezone-aware `datetime` object.

2.1 References

R. G. Allen, L. S. Pereira, D. Raes, and M. Smith, Crop evapotranspiration - Guidelines for computing crop water requirements, FAO Irrigation and drainage paper no. 56, 1998.

CHAPTER 3

History

3.1 1.0.1 (2021-09-01)

- Updated some dependencies, notably htimeseries 3. Python 3.7, 3.8 and 3.9 are now supported.

3.2 1.0.0 (2020-01-05)

- Now uses version 5 of hts file (i.e. different time step notation).

3.3 0.1.2 (2019-10-24)

- Fixed a bug where hts files were opened in the wrong mode, with inconsistent results.

3.4 0.1.1 (2019-07-18)

- Fixed a spurious RuntimeWarning.
- Improved error message if the altitude or location is missing in an input file.

3.5 0.1.0 (2019-06-21)

- Initial release

Index

W