pyeto Documentation

Release 0.2

Mark Richards

Jul 29, 2022

Contents

PyETo is a Python package for calculating reference crop evapotranspiration (ET_o), sometimes referred to as potential evapotranspiration (PET). The library provides numerous functions for estimating missing meteorological data.

Three methods for estimating ET_o/PET are implemented:

- FAO-56 Penman-Monteith (Allen et al, 1998)
- Hargreaves (Hargreaves and Samani, 1985)
- Thornthwaite (Thornthwaite, 1948)

What does it look like? Here is a simple example that estimates monthly potential evapotranspiration for Aberdeen, Scotland (latitude 57.1526 degrees N), using the Thornthwaite method:

```
>>> import pyeto
>>> latitude = pyeto.deg2rad(57.1526)   # Convert latitude to radians
>>> mean_monthly_temperature = [
 >>>     3.1, 3.5, 5.0, 6.7, 9.3, 12.1, 14.3, 14.1, 11.8, 8.9, 5.5, 3.8]
>>> monthly_mean_daylight = pyeto.monthly_mean_daylight_hours(latitude)
>>> pyeto.thornthwaite(mean_monthly_temperature, monthly_mean_daylight)
 [11.04590543317501,
  14.225860424373405,
  27.802870598091953,
  43.178869424774305,
  70.47694909766452,
  93.99420906995957,
  109.69881616481408,
  95.24491684988213,
  64.9945211942068,
  41.06371810827504,
  19.562094545836995,
  12.090183352107148]
```

CHAPTER 1

Installation

There is currently no way to install this package so you will have to include the source in your project in order to use it.

CHAPTER 2

Documentation contents

This part of the documentation guides you through the various evapotranspiration methods.

2.1 Overview

You can install the library directly from PyPI:

```
pip install pyeto
```

Once installed, all public functions and constants are available via the package name:

```
>>> import pyeto
```

PyETo currently implements the following methods for estimating evapotranspiration:

- FAO-56 Penman-Monteith:`pyeto.fao56_penman_monteith()`
- Hargreaves: `pyeto.hargreaves()`
- Thornthwaite: `pyeto.thornthwaite()`

Instructions and examples of using each of these methods is given elsewhere in the documentation.

2.1.1 Estimating missing data

Measurements of the necessary meteorological input variables for each method are frequently not available. To help with this problem, PyETo implements numerous functions for estimating "missing data". Most of these functions are based on the methods described by Allen et al (1998).

For example, atmospheric pressure can be estimated from altitude:

```
>>> pyeto.atm_pressure(1000)  # pressure at 1000 m, in kilo Pascals
90.02461995703662
```

And saturation vapour pressure (*es*), can be estimated from temperature:

```
>>> pyeto.svp_from_t(15.0)  # Sat. vapour pressure in kilo Pascals
1.7053462321157722
```

The API provides details of the functions available for estimating missing data.

2.1.2 Converting units

Careful attention must be paid to the units of each parameter supplied to a function. Different functions may require the same variable, but in different units. To assist with the handling of units, PyETo provides a small collection of functions for converting between commonly used units. For example, a location's latitude in angular degrees can be converted to radians:

```
>>> pyeto.deg2rad(57.1)
0.9965830028887622
```

. . . or from radians into degrees:

```
>>> pyeto.rad2deg(0.9965830028887622)
57.1
```

See the unit conversion section in the API for a full list of unit conversion functions.

2.2 FAO-56 Penman-Monteith method

This is the method recommended by the Food and Agriculture Organisation of the United Nations (FAO) for estimating (ET_o) for a short grass crop using limited meteorological data (see Allen et al, 1998).

The FAO-56 Penman-Monteith equation requires site location, air temperature, humidity, radiation and wind speed data for daily, weekly, ten-day or monthly ET_o calculations. It is important to verify the units of all input data.*PyETo* provides functions to convert common units to the standard unit (see the API of the `pyeto.convert` module).

The instructions given below are a brief summary of those given in Allen et al (1998). It is recommended that you familiarise yourself with chapters 1 to 4 of Allen et al (1998) before proceeding.

2.2.1 Required data

The sections below describe each of the inputs required by the FAO-56 Penman-Monteith equation in `fao.fao56_penman_monteith()`. If measured meteorological data are not available, many of the variables can be estimated using functions in the `fao` module:

```
>>> from pyeto import fao
```

If a measured value is not available for a function's parameter, functions for estimating that parameter are suggested in the function's documentation.

Note, that if monthly $ET_o$is desired, the value of ET_ocalculated with mean monthly weather data is very similar to the average of the daily ET_o values calculated with average weather data for that month.

Location

Altitude above sea level (m) and latitude (degrees north or south) of the location should be specified. These data are needed to adjust some weather parameters for the local average value of atmospheric pressure (a function of the site elevation above mean sea level) and to compute extraterrestrial radiation (Ra) and, in some cases, daylight hours (N). In the calculation procedures for Ra and N, the latitude is expressed in radians (you can use `pyeto.deg2rad()` to convert degrees to radians).

Temperature

The (average) daily maximum and minimum air temperatures in degrees Celsius (°C) are required. Where only (average) mean daily temperatures are available, the calculations can still be executed but some underestimation of ET_o will probably occur due to the non-linearity of the saturation vapour pressure - temperature relationship.

Humidity (vapour pressure)

The (average) daily actual vapour pressure, ea, is required. If measured actual vapour pressure is not available the following functions can be used to estimate actual vapour pressure (in order of preference):

1. If dewpoint temperature data are available use `fao.avp_from_tdew()`.
2. If dry and wet bulb temperatures are available from a psychrometer use `fao.avp_from_twet_tdry()`.
3. If reliable minimum and maximum relative humidity data available use `fao.avp_from_rhmin_rh_max()`.
4. If measurement errors of relative humidity are large then use only maximum relative humidity using `fao.avp_from_rhmax()`
5. If minimum and maximum relative humidity are not available but mean relative humidity is available then use `fao.avp_from_rhmean()` (but this is less reliable than options 3 or 4).
6. If no data for the above are available then use `fao.avp_from_tmin()`. This function is less reliable in arid areas where it is recommended that 2 degrees Celsius is subtracted from the minimum temperature before it is passed to the function (following advice given in Annex 6 of Allen et al (1998).

Saturation vapour pressure (es) is required and can be estimated from air temperature using `fao.svp_from_t()`. The slope of the saturation vapour pressure curve is also required and can be calculated using `fao.svp_from_t()`.

Net radiation

The (average) daily net radiation, Rn, is required. These data are not commonly available but can be derived from the (average) shortwave radiation measured with a pyranometer or from the (average) daily actual duration of bright sunshine (hours per day) measured with a (Campbell-Stokes) sunshine recorder.

Alternatively, if measurements are not available, net radiation can be estimated from net incoming solar (or shortwave) radiation and net outgoing longwave radiation using `fao.net_rad()`.

Net incoming solar radiation

Net solar (or shortwave) radiation is the amount of solar radiation that is not reflected by the surface and can be calculated using `fao.net_in_sol_rad()`.

Solar (shortwave) radiation

The amount of incoming solar radiation (or shortwave radiation) reaching a horizontal plane after scattering by the atmosphere. If measured values of gross incoming solar radiation are not available the following functions (in order of preference), can be used to estimate it:

1. If sunshine duration data are available use `fao.sol_rad_from_sun_hours()`.
2. Otherwise use `fao.sol_rad_from_t()` which requires minimum and maximum temperature. Suitable for coastal or inland areas but not islands.
3. For island locations (<= 20 km wide), where no measured values are available from elsewhere on the island and the altitude is 0-100 m, use `fao.sol_rad_island()`. Only suitable for monthly calculations.

Net outgoing longwave radiation

Net outgoing longwave radiation is the net longwave energy leaving the earth's surface. It is proportional to the absolute temperature of the surface raised to the fourth power according to the Stefan-Boltzmann law. However, water vapour, clouds, carbon dioxide and dust are absorbers and emitters of longwave radiation. This function corrects the Stefan-Boltzmann law for humidity (using actual vapor pressure) and cloudiness (using solar radiation and clear sky radiation). Net outgoing longwave radiation can be estimated using `fao.net_out_lw_rad()`.

Psychrometric constant

The psychrometric constant is the ratio of specific heat of moist air at constant pressure to latent heat of vaporization of water. It can be estimated from atmospheric pressure using `fao.psy_const()` or `fao.psy_const_of_psychrometer()`.

Soil heat flux

For a daily time step soil heat flux is small compared to net radiation when the soil is covered by vegetation, so it can be assumed to be zero (Allen et al, 1998).

For a monthly time step soil heat flux is significant and should be estimated using:

1. `fao.monthly_soil_heat_flux()` if temperature data for the previous and next month is available, or
2. `fao.monthly_soil_heat_flux2()` if temperature for the next month is not available.

2.3 Hargreaves equation

The Hargreaves equation (Hargreaves and Samani, 1985) is a simple evapotranspiration model that only requires a few easily accessible parameters: mininimum, maximum and mean temperature, and extraterrestrial radiation.

The Hargreaves method is recommended by the FAO (Allen et al, 1998) as an alternative method for estimating ETo if insufficient meteorological data are available for the Penman-Monteith method. However, the FAO suggest that using the Penman-Monteith method with estimated solar radiation, vapor pressure and wind speed generally provides more accurate estimates than the Hargreaves equation. This is due to the ability of the estimation equations to incorporate general climatic characteristics such as high or low wind speed or high or low relative humidity into the ET_o estimate made using the FAO Penman-Monteith method.

The Hargreaves equation has a tendency to under-estimate ET_ounder high wind conditions(u2 > 3m/s) and to over-estimate under conditions of high relative humidity.

The following example uses the Hargreaves model to estimate monthly PET for the 1st of February, 2014, for Aberdeen, Scotland (latitude 57.1526 degrees N).

First, convert latitude to radians and the date to day of the year (Julian day):

```
>>> import datetime, pyeto
>>> lat = pyeto.deg2rad(57.1526)  # Convert latitude to radians
>>> day_of_year = datetime.date(2014, 2, 1).timetuple().tm_yday
```

To estimate extraterrestrial radiation we first need to calculate solar declination, sunset hour angle and inverse relative distance Earth-Sun:

```
>>> sol_dec = pyeto.sol_dec(day_of_year)            # Solar declination
>>> sha = pyeto.sunset_hour_angle(lat, sol_dec)
>>> ird = pyeto.inv_rel_dist_earth_sun(day_of_year)
>>> et_rad = pyeto.et_rad(lat, sol_dec, sha, ird)   # Extraterrestrial radiation
```

Finally, we can estimate ET_o, assuming a minimum temperature of 1.3, maximum temperature of 5.6 and mean temperature of 3.8:

```
>>> hargreaves(1.3, 5.6, 3.8, et_rad)
```

2.4 Thornthwaite equation

The Thornthwaite (1948) equation is a widely used empirical method for estimating potential evapotranspiration (PET). The equation only requires mean monthly air temperature and mean daily daylight hours for each month, which can be calculated from latitude.

The following example estimates monthly PET in 2014 for Aberdeen, Scotland (latitude 57.1526 degrees N):

First convert latitude to radians and calculate the monthly mean daylight hours:

```
>>> from pyeto import thornthwaite, monthly_mean_daylight_hours, deg2rad
>>> lat = deg2rad(57.1526)  # Convert latitude in degrees to radians
>>> # Calculate mean daylight hours of each month
>>> mmdlh = monthly_mean_daylight_hours(lat, 2014)
>>> mmdlh
[7.182842574993897,
 9.13512841264262,
 11.523002734356053,
 14.035348256722466,
 16.277003584884323,
 17.505213218539176,
 16.891449464611544,
 14.861767363416547,
 12.394150156712453,
 9.88498386070613,
 7.658250142104072,
 6.489516585536734]
>>> # Create iterable of monthly mean temperatures in degrees Celsius
>>> monthly_t = [
>>>     3.1, 3.5, 5.0, 6.7, 9.3, 12.1, 14.3, 14.1, 11.8, 8.9, 5.5, 3.8]
>>> thornthwaite(monthly_t, mmdlh)   # Calculate PET
[11.04590543317501,
 14.225860424373405,
 27.802870598091953,
```

(continues on next page)

(continued from previous page)

```
43.178869424774305,
70.47694909766452,
93.99420906995957,
109.69881616481408,
95.24491684988213,
64.9945211942068,
41.06371810827504,
19.562094545836995,
12.090183352107148]
```

2.5 References

Allen RG, Pereira LS, Raes D, Smith M (1998) Crop evapotranspiration. Guidelines for computing crop water requirements. FAO irrigation and drainage paper 56, FAO, Rome.

Hargreaves GH, Samani ZA (1982) Estimating potential evapotranspiration. Journal of the Irrigation and Drainage Division, ASCE, 108(3):225-230.

Hargreaves GH, Samani ZA (1985) Reference crop evapotranspiration from temperature. Applied Engineering in Agriculture 1(2):96-99 (doi 10.13031/2013.26773).

Thornthwaite CW (1948) An approach toward a rational classification of climate. Geographical Review, 38, 55-94.

2.6 Changelog

2.6.1 Version 1.0rc1

(first release candidate for PyETo 1.0, released on TODO:insert date here)

- Initial release.

2.7 License

PyETo is licensed under the BSD 3-Clause "New" or "Revised" License.

2.7.1 License text

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHAPTER 3

API Reference

If you are looking for information on a specific function, this part of the documentation is for you.

3.1 API

This part of the documentation shows the full API reference of all public functions.

3.1.1 Evapotranspiration

pyeto.**fao56_penman_monteith**(*net_rad*, *t*, *ws*, *svp*, *avp*, *delta_svp*, *psy*, *shf=0.0*)

Estimate reference evapotranspiration (ETo) from a hypothetical short grass reference surface using the FAO-56 Penman-Monteith equation.

Based on equation 6 in Allen et al (1998).

Parameters

- **net_rad** – Net radiation at crop surface [MJ m-2 day-1]. If necessary this can be estimated using `net_rad()`.
- **t** – Air temperature at 2 m height [deg Kelvin].
- **ws** – Wind speed at 2 m height [m s-1]. If not measured at 2m, convert using `wind_speed_at_2m()`.
- **svp** – Saturation vapour pressure [kPa]. Can be estimated using ‘‘svp_from_t()”.
- **avp** – Actual vapour pressure [kPa]. Can be estimated using a range of functions with names beginning with ‘avp_from’.
- **delta_svp** – Slope of saturation vapour pressure curve [kPa degC-1]. Can be estimated using `delta_svp()`.
- **psy** – Psychrometric constant [kPa deg C]. Can be estimatred using `psy_const_of_psychrometer()` or `psy_const()`.

- **shf** – Soil heat flux (G) [MJ m-2 day-1] (default is 0.0, which is reasonable for a daily or 10-day time steps). For monthly time steps *shf* can be estimated using `monthly_soil_heat_flux()` or `monthly_soil_heat_flux2()`.

Returns Reference evapotranspiration (ETo) from a hypothetical grass reference surface [mm day-1].

Return type float

`pyeto.`**`hargreaves`**(*tmin*, *tmax*, *tmean*, *et_rad*)

Estimate reference evapotranspiration over grass (ETo) using the Hargreaves equation.

Generally, when solar radiation data, relative humidity data and/or wind speed data are missing, it is better to estimate them using the functions available in this module, and then calculate ETo the FAO Penman-Monteith equation. However, as an alternative, ETo can be estimated using the Hargreaves ETo equation.

Based on equation 52 in Allen et al (1998).

Parameters

- **tmin** – Minimum daily temperature [deg C]
- **tmax** – Maximum daily temperature [deg C]
- **tmean** – Mean daily temperature [deg C]. If emasurements not available it can be estimated as (*tmin* + *tmax*) / 2.
- **et_rad** – Extraterrestrial radiation (Ra) [MJ m-2 day-1]. Can be estimated using `et_rad()`.

Returns Reference evapotranspiration over grass (ETo) [mm day-1]

Return type float

`pyeto.`**`thornthwaite`**(*monthly_t*, *monthly_mean_dlh*, *year=None*)

Estimate monthly potential evapotranspiration (PET) using the Thornthwaite (1948) method.

Thornthwaite equation:

PET = 1.6 (*L*/12) (*N*/30) (10*Ta* / *I*)***a*

where:

- *Ta* is the mean daily air temperature [deg C, if negative use 0] of the month being calculated
- *N* is the number of days in the month being calculated
- *L* is the mean day length [hours] of the month being calculated
- *a* = (6.75 x 10-7)*I***3 - (7.71 x 10-5)*I***2 + (1.792 x 10-2)*I* + 0.49239
- *I* is a heat index which depends on the 12 monthly mean temperatures and is calculated as the sum of (*Tai* / 5)**1.514 for each month, where Tai is the air temperature for each month in the year

Parameters

- **monthly_t** – Iterable containing mean daily air temperature for each month of the year [deg C].
- **monthly_mean_dlh** – Iterable containing mean daily daylight hours for each month of the year (hours]. These can be calculated using `monthly_mean_daylight_hours()`.
- **year** – Year for which PET is required. The only effect of year is to change the number of days in February to 29 if it is a leap year. If it is left as the default (None), then the year is assumed not to be a leap year.

Returns Estimated monthly potential evaporation of each month of the year [mm/month]

Return type List of floats

3.1.2 Atmospheric pressure

pyeto.**atm_pressure**(*altitude*)
Estimate atmospheric pressure from altitude.

Calculated using a simplification of the ideal gas law, assuming 20 degrees Celsius for a standard atmosphere. Based on equation 7, page 62 in Allen et al (1998).

Parameters **altitude** – Elevation/altitude above sea level [m]

Returns atmospheric pressure [kPa]

Return type float

3.1.3 Daylight hours

pyeto.**daylight_hours**(*sha*)
Calculate daylight hours from sunset hour angle.

Based on FAO equation 34 in Allen et al (1998).

Parameters **sha** – Sunset hour angle [rad]. Can be calculated using `sunset_hour_angle()`.

Returns Daylight hours.

Return type float

pyeto.**monthly_mean_daylight_hours**(*latitude*, *year=None*)
Calculate mean daylight hours for each month of the year for a given latitude.

Parameters

- **latitude** – Latitude [radians]
- **year** – Year for the daylight hours are required. The only effect of *year* is to change the number of days in Feb to 29 if it is a leap year. If left as the default, None, then a normal (non-leap) year is assumed.

Returns Mean daily daylight hours of each month of a year [hours]

Return type List of floats.

3.1.4 Humidity

Actual vapour pressure (ea)

pyeto.**avp_from_rhmax**(*svp_tmin*, *rh_max*)
Estimate actual vapour pressure (*e*a) from saturation vapour pressure at daily minimum temperature and maximum relative humidity

Based on FAO equation 18 in Allen et al (1998).

Parameters

- **svp_tmin** – Saturation vapour pressure at daily minimum temperature [kPa]. Can be estimated using `svp_from_t()`.

- **rh_max** – Maximum relative humidity [%]

Returns Actual vapour pressure [kPa]

Return type float

pyeto.**avp_from_rhmean**(*svp_tmin*, *svp_tmax*, *rh_mean*)
Estimate actual vapour pressure (*ea*) from saturation vapour pressure at daily minimum and maximum temperature, and mean relative humidity.

Based on FAO equation 19 in Allen et al (1998).

Parameters

- **svp_tmin** – Saturation vapour pressure at daily minimum temperature [kPa]. Can be estimated using `svp_from_t()`.
- **svp_tmax** – Saturation vapour pressure at daily maximum temperature [kPa]. Can be estimated using `svp_from_t()`.
- **rh_mean** – Mean relative humidity [%] (average of RH min and RH max).

Returns Actual vapour pressure [kPa]

Return type float

pyeto.**avp_from_rhmin_rhmax**(*svp_tmin*, *svp_tmax*, *rh_min*, *rh_max*)
Estimate actual vapour pressure (*ea*) from saturation vapour pressure and relative humidity.

Based on FAO equation 17 in Allen et al (1998).

Parameters

- **svp_tmin** – Saturation vapour pressure at daily minimum temperature [kPa]. Can be estimated using `svp_from_t()`.
- **svp_tmax** – Saturation vapour pressure at daily maximum temperature [kPa]. Can be estimated using `svp_from_t()`.
- **rh_min** – Minimum relative humidity [%]
- **rh_max** – Maximum relative humidity [%]

Returns Actual vapour pressure [kPa]

Return type float

pyeto.**avp_from_tdew**(*tdew*)
Estimate actual vapour pressure (*ea*) from dewpoint temperature.

Based on equation 14 in Allen et al (1998). As the dewpoint temperature is the temperature to which air needs to be cooled to make it saturated, the actual vapour pressure is the saturation vapour pressure at the dewpoint temperature.

This method is preferable to calculating vapour pressure from minimum temperature.

Parameters **tdew** – Dewpoint temperature [deg C]

Returns Actual vapour pressure [kPa]

Return type float

pyeto.**avp_from_tmin**(*tmin*)
Estimate actual vapour pressure (*ea*) from minimum temperature.

This method is to be used where humidity data are lacking or are of questionable quality. The method assumes that the dewpoint temperature is approximately equal to the minimum temperature (*tmin*), i.e. the air is saturated with water vapour at *tmin*.

Note: This assumption may not hold in arid/semi-arid areas. In these areas it may be better to subtract 2 deg C from the minimum temperature (see Annex 6 in FAO paper).

Based on equation 48 in Allen et al (1998).

Parameters **tmin** – Daily minimum temperature [deg C]

Returns Actual vapour pressure [kPa]

Return type float

pyeto.**avp_from_twet_tdry**(*twet*, *tdry*, *svp_twet*, *psy_const*)

Estimate actual vapour pressure (*ea*) from wet and dry bulb temperature.

Based on equation 15 in Allen et al (1998). As the dewpoint temperature is the temperature to which air needs to be cooled to make it saturated, the actual vapour pressure is the saturation vapour pressure at the dewpoint temperature.

This method is preferable to calculating vapour pressure from minimum temperature.

Values for the psychrometric constant of the psychrometer (*psy_const*) can be calculated using `psyc_const_of_psychrometer()`.

Parameters

- **twet** – Wet bulb temperature [deg C]
- **tdry** – Dry bulb temperature [deg C]
- **svp_twet** – Saturated vapour pressure at the wet bulb temperature [kPa]. Can be estimated using `svp_from_t()`.
- **psy_const** – Psychrometric constant of the pyschrometer [kPa deg C-1]. Can be estimated using `psy_const()` or `psy_const_of_psychrometer()`.

Returns Actual vapour pressure [kPa]

Return type float

Saturated vapour pressure (es)

pyeto.**delta_svp**(*t*)

Estimate the slope of the saturation vapour pressure curve at a given temperature.

Based on equation 13 in Allen et al (1998). If using in the Penman-Monteith *t* should be the mean air temperature.

Parameters **t** – Air temperature [deg C]. Use mean air temperature for use in Penman-Monteith.

Returns Saturation vapour pressure [kPa degC-1]

Return type float

pyeto.**mean_svp**(*tmin*, *tmax*)

Estimate mean saturation vapour pressure, *es* [kPa] from minimum and maximum temperature.

Based on equations 11 and 12 in Allen et al (1998).

Mean saturation vapour pressure is calculated as the mean of the saturation vapour pressure at tmax (maximum temperature) and tmin (minimum temperature).

Parameters

- **tmin** – Minimum temperature [deg C]
- **tmax** – Maximum temperature [deg C]

Returns Mean saturation vapour pressure (*es*) [kPa]

Return type float

pyeto.**svp_from_t**(*t*)
Estimate saturation vapour pressure (*es*) from air temperature.

Based on equations 11 and 12 in Allen et al (1998).

Parameters **t** – Temperature [deg C]

Returns Saturation vapour pressure [kPa]

Return type float

Relative humidity (RH)

pyeto.**rh_from_avp_svp**(*avp*, *svp*)
Calculate relative humidity as the ratio of actual vapour pressure to saturation vapour pressure at the same temperature.

See Allen et al (1998), page 67 for details.

Parameters

- **avp** – Actual vapour pressure [units do not matter so long as they are the same as for *svp*]. Can be estimated using functions whose name begins with 'avp_from'.
- **svp** – Saturated vapour pressure [units do not matter so long as they are the same as for *avp*]. Can be estimated using `svp_from_t()`.

Returns Relative humidity [%].

Return type float

3.1.5 Pyschrometric constant

pyeto.**psy_const**(*atmos_pres*)
Calculate the psychrometric constant.

This method assumes that the air is saturated with water vapour at the minimum daily temperature. This assumption may not hold in arid areas.

Based on equation 8, page 95 in Allen et al (1998).

Parameters **atmos_pres** – Atmospheric pressure [kPa]. Can be estimated using `atm_pressure()`.

Returns Psychrometric constant [kPa degC-1].

Return type float

pyeto.**psy_const_of_psychrometer**(*psychrometer*, *atmos_pres*)
Calculate the psychrometric constant for different types of psychrometer at a given atmospheric pressure.

Based on FAO equation 16 in Allen et al (1998).

Parameters

- **psychrometer** – Integer between 1 and 3 which denotes type of psychrometer: 1. ventilated (Asmann or aspirated type) psychrometer with

  an air movement of approximately 5 m/s

  2. natural ventilated psychrometer with an air movement of approximately 1 m/s
  3. non ventilated psychrometer installed indoors

- **atmos_pres** – Atmospheric pressure [kPa]. Can be estimated using `atm_pressure()`.

Returns Psychrometric constant [kPa degC-1].

Return type float

3.1.6 Radiation

`pyeto.`**`cs_rad`**(*altitude*, *et_rad*)

Estimate clear sky radiation from altitude and extraterrestrial radiation.

Based on equation 37 in Allen et al (1998) which is recommended when calibrated Angstrom values are not available.

Parameters

- **altitude** – Elevation above sea level [m]
- **et_rad** – Extraterrestrial radiation [MJ m-2 day-1]. Can be estimated using `et_rad()`.

Returns Clear sky radiation [MJ m-2 day-1]

Return type float

`pyeto.`**`et_rad`**(*latitude*, *sol_dec*, *sha*, *ird*)

Estimate daily extraterrestrial radiation (*Ra*, 'top of the atmosphere radiation').

Based on equation 21 in Allen et al (1998). If monthly mean radiation is required make sure *sol_dec*. *sha* and *irl* have been calculated using the day of the year that corresponds to the middle of the month.

Note: From Allen et al (1998): "For the winter months in latitudes greater than 55 degrees (N or S), the equations have limited validity. Reference should be made to the Smithsonian Tables to assess possible deviations."

Parameters

- **latitude** – Latitude [radians]
- **sol_dec** – Solar declination [radians]. Can be calculated using `sol_dec()`.
- **sha** – Sunset hour angle [radians]. Can be calculated using `sunset_hour_angle()`.
- **ird** – Inverse relative distance earth-sun [dimensionless]. Can be calculated using `inv_rel_dist_earth_sun()`.

Returns Daily extraterrestrial radiation [MJ m-2 day-1]

Return type float

`pyeto.`**`net_in_sol_rad`**(*sol_rad*, *albedo=0.23*)

Calculate net incoming solar (or shortwave) radiation from gross incoming solar radiation, assuming a grass reference crop.

Net incoming solar radiation is the net shortwave radiation resulting from the balance between incoming and reflected solar radiation. The output can be converted to equivalent evaporation [mm day-1] using `energy2evap()`.

Based on FAO equation 38 in Allen et al (1998).

Parameters

- **sol_rad** – Gross incoming solar radiation [MJ m-2 day-1]. If necessary this can be estimated using functions whose name begins with 'sol_rad_from'.
- **albedo** – Albedo of the crop as the proportion of gross incoming solar radiation that is reflected by the surface. Default value is 0.23, which is the value used by the FAO for a short grass reference crop. Albedo can be as high as 0.95 for freshly fallen snow and as low as 0.05 for wet bare soil. A green vegetation over has an albedo of about 0.20-0.25 (Allen et al, 1998).

Returns Net incoming solar (or shortwave) radiation [MJ m-2 day-1].

Return type float

pyeto.**net_out_lw_rad**(*tmin*, *tmax*, *sol_rad*, *cs_rad*, *avp*)

Estimate net outgoing longwave radiation.

This is the net longwave energy (net energy flux) leaving the earth's surface. It is proportional to the absolute temperature of the surface raised to the fourth power according to the Stefan-Boltzmann law. However, water vapour, clouds, carbon dioxide and dust are absorbers and emitters of longwave radiation. This function corrects the Stefan- Boltzmann law for humidity (using actual vapor pressure) and cloudiness (using solar radiation and clear sky radiation). The concentrations of all other absorbers are assumed to be constant.

The output can be converted to equivalent evaporation [mm day-1] using `energy2evap()`.

Based on FAO equation 39 in Allen et al (1998).

Parameters

- **tmin** – Absolute daily minimum temperature [degrees Kelvin]
- **tmax** – Absolute daily maximum temperature [degrees Kelvin]
- **sol_rad** – Solar radiation [MJ m-2 day-1]. If necessary this can be estimated using `sol+rad()`.
- **cs_rad** – Clear sky radiation [MJ m-2 day-1]. Can be estimated using `cs_rad()`.
- **avp** – Actual vapour pressure [kPa]. Can be estimated using functions with names beginning with 'avp_from'.

Returns Net outgoing longwave radiation [MJ m-2 day-1]

Return type float

pyeto.**net_rad**(*ni_sw_rad*, *no_lw_rad*)

Calculate daily net radiation at the crop surface, assuming a grass reference crop.

Net radiation is the difference between the incoming net shortwave (or solar) radiation and the outgoing net longwave radiation. Output can be converted to equivalent evaporation [mm day-1] using `energy2evap()`.

Based on equation 40 in Allen et al (1998).

Parameters

- **ni_sw_rad** – Net incoming shortwave radiation [MJ m-2 day-1]. Can be estimated using `net_in_sol_rad()`.
- **no_lw_rad** – Net outgoing longwave radiation [MJ m-2 day-1]. Can be estimated using `net_out_lw_rad()`.

Returns Daily net radiation [MJ m-2 day-1].

Return type float

pyeto.**sol_rad_from_sun_hours**(*daylight_hours*, *sunshine_hours*, *et_rad*)

Calculate incoming solar (or shortwave) radiation, *Rs* (radiation hitting a horizontal plane after scattering by the atmosphere) from relative sunshine duration.

If measured radiation data are not available this method is preferable to calculating solar radiation from temperature. If a monthly mean is required then divide the monthly number of sunshine hours by number of days in the month and ensure that *et_rad* and *daylight_hours* was calculated using the day of the year that corresponds to the middle of the month.

Based on equations 34 and 35 in Allen et al (1998).

Parameters

- **dl_hours** – Number of daylight hours [hours]. Can be calculated using `daylight_hours()`.
- **sunshine_hours** – Sunshine duration [hours].
- **et_rad** – Extraterrestrial radiation [MJ m-2 day-1]. Can be estimated using `et_rad()`.

Returns Incoming solar (or shortwave) radiation [MJ m-2 day-1]

Return type float

pyeto.**sol_rad_from_t**(*et_rad*, *cs_rad*, *tmin*, *tmax*, *coastal*)

Estimate incoming solar (or shortwave) radiation, *Rs*, (radiation hitting a horizontal plane after scattering by the atmosphere) from min and max temperature together with an empirical adjustment coefficient for 'interior' and 'coastal' regions.

The formula is based on equation 50 in Allen et al (1998) which is the Hargreaves radiation formula (Hargreaves and Samani, 1982, 1985). This method should be used only when solar radiation or sunshine hours data are not available. It is only recommended for locations where it is not possible to use radiation data from a regional station (either because climate conditions are heterogeneous or data are lacking).

NOTE: this method is not suitable for island locations due to the moderating effects of the surrounding water.

Parameters

- **et_rad** – Extraterrestrial radiation [MJ m-2 day-1]. Can be estimated using `et_rad()`.
- **cs_rad** – Clear sky radiation [MJ m-2 day-1]. Can be estimated using `cs_rad()`.
- **tmin** – Daily minimum temperature [deg C].
- **tmax** – Daily maximum temperature [deg C].
- **coastal** – `True` if site is a coastal location, situated on or adjacent to coast of a large land mass and where air masses are influenced by a nearby water body, `False` if interior location where land mass dominates and air masses are not strongly influenced by a large water body.

Returns Incoming solar (or shortwave) radiation (Rs) [MJ m-2 day-1].

Return type float

pyeto.**sol_rad_island**(*et_rad*)

Estimate incoming solar (or shortwave) radiation, *Rs* (radiation hitting a horizontal plane after scattering by the atmosphere) for an island location.

An island is defined as a land mass with width perpendicular to the coastline <= 20 km. Use this method only if radiation data from elsewhere on the island is not available.

NOTE: This method is only applicable for low altitudes (0-100 m) and monthly calculations.

Based on FAO equation 51 in Allen et al (1998).

Parameters **et_rad** – Extraterrestrial radiation [MJ m-2 day-1]. Can be estimated using `et_rad()`.

Returns Incoming solar (or shortwave) radiation [MJ m-2 day-1].

Return type float

3.1.7 Soil heat flux

`pyeto.`**monthly_soil_heat_flux**(*t_month_prev*, *t_month_next*)

Estimate monthly soil heat flux (Gmonth) from the mean air temperature of the previous and next month, assuming a grass crop.

Based on equation 43 in Allen et al (1998). If the air temperature of the next month is not known use `monthly_soil_heat_flux2()` instead. The resulting heat flux can be converted to equivalent evaporation [mm day-1] using `energy2evap()`.

Parameters

- **t_month_prev** – Mean air temperature of the previous month [deg Celsius]
- **t_month2_next** – Mean air temperature of the next month [deg Celsius]

Returns Monthly soil heat flux (Gmonth) [MJ m-2 day-1]

Return type float

`pyeto.`**monthly_soil_heat_flux2**(*t_month_prev*, *t_month_cur*)

Estimate monthly soil heat flux (Gmonth) [MJ m-2 day-1] from the mean air temperature of the previous and current month, assuming a grass crop.

Based on equation 44 in Allen et al (1998). If the air temperature of the next month is available, use `monthly_soil_heat_flux()` instead. The resulting heat flux can be converted to equivalent evaporation [mm day-1] using `energy2evap()`.

Arguments: :param t_month_prev: Mean air temperature of the previous month

[deg Celsius]

Parameters **t_month_cur** – Mean air temperature of the current month [deg Celsius]

Returns Monthly soil heat flux (Gmonth) [MJ m-2 day-1]

Return type float

3.1.8 Solar angles etc.

`pyeto.`**inv_rel_dist_earth_sun**(*day_of_year*)

Calculate the inverse relative distance between earth and sun from day of the year.

Based on FAO equation 23 in Allen et al (1998).

Parameters **day_of_year** – Day of the year [1 to 366]

Returns Inverse relative distance between earth and the sun

Return type float

pyeto.**sol_dec**(*day_of_year*)
Calculate solar declination from day of the year.

Based on FAO equation 24 in Allen et al (1998).

Parameters **day_of_year** – Day of year integer between 1 and 365 or 366).

Returns solar declination [radians]

Return type float

pyeto.**sunset_hour_angle**(*latitude*, *sol_dec*)
Calculate sunset hour angle (*Ws*) from latitude and solar declination.

Based on FAO equation 25 in Allen et al (1998).

Parameters

- **latitude** – Latitude [radians]. Note: *latitude* should be negative if it in the southern hemisphere, positive if in the northern hemisphere.
- **sol_dec** – Solar declination [radians]. Can be calculated using `sol_dec()`.

Returns Sunset hour angle [radians].

Return type float

3.1.9 Temperature

pyeto.**daily_mean_t**(*tmin*, *tmax*)
Estimate mean daily temperature from the daily minimum and maximum temperatures.

Parameters

- **tmin** – Minimum daily temperature [deg C]
- **tmax** – Maximum daily temperature [deg C]

Returns Mean daily temperature [deg C]

Return type float

3.1.10 Wind speed

pyeto.**wind_speed_2m**(*ws*, *z*)
Convert wind speed measured at different heights above the soil surface to wind speed at 2 m above the surface, assuming a short grass surface.

Based on FAO equation 47 in Allen et al (1998).

Parameters

- **ws** – Measured wind speed [m s-1]
- **z** – Height of wind measurement above ground surface [m]

Returns Wind speed at 2 m above the surface [m s-1]

Return type float

3.1.11 Constants

pyeto.fao.**SOLAR_CONSTANT = 0.082**
Solar constant [MJ m-2 min-1]

pyeto.fao.**STEFAN_BOLTZMANN_CONSTANT = 4.903e-09**
Stefan Boltzmann constant [MJ K-4 m-2 day-1]

3.1.12 Unit conversion

pyeto.**celsius2kelvin**(*celsius*)
Convert temperature in degrees Celsius to degrees Kelvin.

Parameters **celsius** – Degrees Celsius

Returns Degrees Kelvin

Return type float

pyeto.**deg2rad**(*degrees*)
Convert angular degrees to radians

Parameters **degrees** – Value in degrees to be converted.

Returns Value in radians

Return type float

pyeto.**energy2evap**(*energy*)
Convert energy (e.g. radiation energy) in MJ m-2 day-1 to the equivalent evaporation, assuming a grass reference crop.

Energy is converted to equivalent evaporation using a conversion factor equal to the inverse of the latent heat of vapourisation (1 / lambda = 0.408).

Based on FAO equation 20 in Allen et al (1998).

Parameters **energy** – Energy e.g. radiation or heat flux [MJ m-2 day-1].

Returns Equivalent evaporation [mm day-1].

Return type float

pyeto.**kelvin2celsius**(*kelvin*)
Convert temperature in degrees Kelvin to degrees Celsius.

Parameters **kelvin** – Degrees Kelvin

Returns Degrees Celsius

Return type float

pyeto.**rad2deg**(*radians*)
Convert radians to angular degrees

Parameters **radians** – Value in radians to be converted.

Returns Value in angular degrees

Return type float

Python Module Index

p

Index