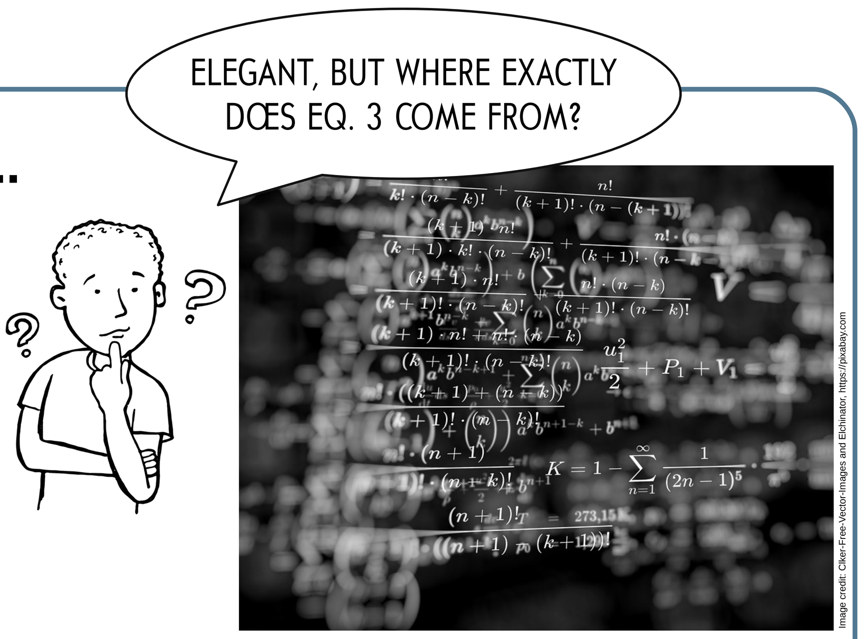
Open and reproducible science: From theory to equations, algorithms, and plots

Stan Schymanski (1), Jiří Kunčar (2, 3)

(1) Environmental Research and Innovation (ERIN), Luxembourg Institute of Science and Technology, Belvaux, Luxembourg
(2) Swiss Data Science Center (SDSC), Zurich, Switzerland
(3) Datadog Inc.

Motivation



Mathematical expressions are...

- the result of scientific theory
- the starting point of new theory
- carriers of explicit and implicit assumptions
- often of unclear origin
- sometimes void of meaning
- at the base of quantitative computations and predictions

→ **Need an open computational framework to transparently link data to algorithms, equations and their underlying theory and assumptions.**

Open-source: python, SymPy, ESSM and jupyter

python.org

"Python is a programming language that lets you work more quickly and integrate your systems more effectively."
- Transparent, free and open source
- Well documented and easy to learn
- Interactive, inclusive
- Community-developed, extendable

sympy.org

"SymPy is a Python library for symbolic mathematics."
- Transparent, free and open source
- Solve systems of equations and inequalities symbolically
- Step-by-step manipulations of expressions
- Plethora of mathematical solvers and operations (differentials, integrals, summations, factorials, ...)

github.com/environmentalscience/essm

"Environmental Science using Symbolic Math (ESSM) contains helpers to deal with physical variables and units."
- Transparent, free and open source
- Record dimensions, units and standard values with variable definitions
- Record dependencies between expressions
- Automatically check for dimensional consistency

jupyter.org

"The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text."
- Transparent, free and open source
- Interactive
- Easy to share and collaborate

Example: Unit mystery in PT equation

The Priestley-Taylor (PT) equation (Priestley & Taylor, 1972) for large scale evaporation is widely used, yet its derivation is not entirely clear.

A key step in the derivation is Equation 3:

> for all time. It had been earlier concluded by one of the authors (Priestley 1959) that the solution would then be $\psi=0$ for all t and z and thus
>
> $$\frac{LE}{H}=\frac{L}{c_p}\left(\frac{\partial q_s}{\partial T}\right)_{T=\overline{T}}=\frac{s}{\gamma}, \qquad (3)$$
>
> c_p is the specific heat of air at constant pressure, s is defined as $\partial q_s/\partial T$ at the appropriate temperature, and γ is c_p/L.

Since the units of the variables were not specified in the paper, we make **informed guesses** based on the description in the text and **widespread literature conventions**. The resulting variables in a Jupyter Notebook:

In [7]:
```python
generate_metadata_table([E_w, H_w, L_E, c_pa, q_s, s_PT, gamma_PT])
```

Out[7]:

Symbol	Name	Description	Definition	Default value	Units
γ	gamma_PT	Priestley-Taylor gamma = c_p/L	$\frac{c_{pa}}{L}$	-	K^{-1}
c_{pa}	c_pa	Specific heat of dry air.		1010.0	$J K^{-1} kg^{-1}$
E	E_w	Wet surface evaporation (E in Priestley and Taylor, 1972))		-	$kg s^{-1} m^{-2}$
H	H_w	Wet surface sensible heat flux (positive outwards)		-	$W m^{-2}$
L	L_E	Latent heat		2450000.0	$J kg^{-1}$
q_s	q_s	Specific humidity at saturation		-	$kg m^{-3}$
s	s_PT	Priestley-Taylor $\partial q_s/\partial T$	$\frac{d}{dT} q_s$	-	$kg K^{-1} m^{-3}$

Entering Eq. 3 as a physical equation in essm returns error:

In [8]:
```python
eq3 = Eq(L_E*E_w/H_w, L_E/c_pa*s_PT)
display(eq3)
try:
    class eq_3(Equation):
        """Equation 3 in Priestley and Taylor (1972)"""
        expr = eq3
except Exception as error:
    print(error)
```

$$\frac{EL}{H} = \frac{Ls}{c_{pa}}$$

```
Dimension of "L_E*s_PT/c_pa" is Dimension(mass/length**3), but it should be the
same as E_w*L_E/H_w, i.e. Dimension(1)
```

Indeed, the left-hand-side of Eq. 3 is dimensionless, while the right-hand side is not:

In [9]:
```python
derive_baseunit(eq3.rhs)
```

Out[9]: $\frac{kg}{m^3}$

No amount of guessing led to dimensional consistency of Eq. 3 and the problem carries through the whole paper, as *γ* and *s* must, but do not have the same units, e.g. Eq. 5:

In [29]:
```python
alpha = Symbol('alpha')
eq5 = Eq(LE / (LE + H_w), alpha * s_PT/(s_PT + gamma_PT))
display(eq5)
try: derive_baseunit(eq5.rhs)
except Exception as e1: print(e1)
```

$$\frac{LE}{H + LE} = \frac{s\alpha}{\gamma + s}$$

```
Dimension of "s_PT" is Dimension(mass/(length**3*temperature)), but it should be
the same as gamma_PT, i.e. Dimension(1/temperature)
```

Example: PT vs. Penman equation

The PT equation is based on a derivation by Penman (1948), who used different variable definitions:

In [31]:
```python
generate_metadata_table([Delta_eTa, gamma_P])
```

Out[31]:

Symbol	Name	Description	Definition	Default value	Units
Δ	Delta_eTa	Slope of saturation vapour pressure at air temperature		-	$Pa K^{-1}$
γ_P	gamma_P	Penman gamma, gamma = beta(e_s - e_d)/(T_s - T_a)	$\frac{\beta(-e_d+e_s)}{-T_a+T_s}$	-	$Pa K^{-1}$

We can easily **substitute** Penman's variables and the energy balance into the original PT Eq. 5, **solve** for LE, and **plot** the resulting PT equation against the original Penman equation:

In [32]:
```python
eq5P = eq5.subs({s_PT: Delta_eTa, gamma_PT: gamma_P}); display(eq5P)
derive_unit(eq5P.rhs)
```

$$\frac{LE}{H + LE} = \frac{\Delta\alpha}{\Delta + \gamma_P}$$

Out[32]: 1

Finally consistent units!

In [34]:
```python
class eq_enbalance(Equation):
    """Surface energy balance."""
    expr = Eq(R - G, LE + H_w)
eq_enbalance
```

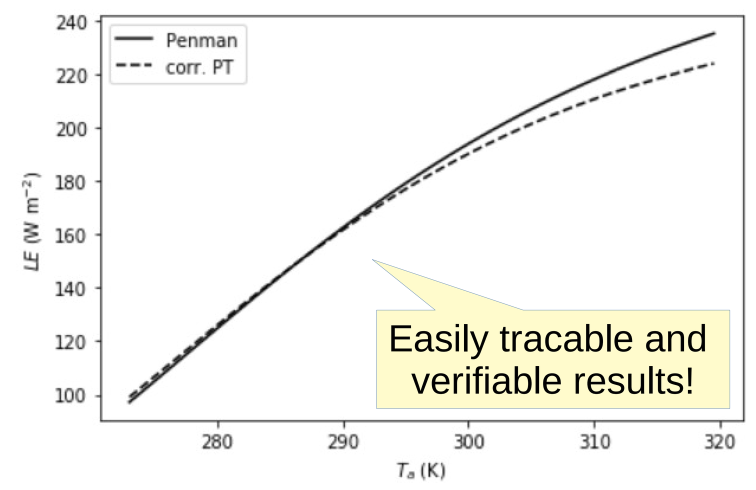
Out[34]: $-G + R = H + LE$

In [35]:
```python
class eq_8P(eq_enbalance.definition, eq_5P.definition):
    """Eq. 8 in Priestley & Taylor (1972) using Penman's variable definitions
    soln = solve([eq_5P, eq_enbalance], LE, H_w)
    expr = Eq(LE, soln[LE])
eq_8P
```

Out[35]: $LE = -\frac{\Delta\alpha(G - R)}{\Delta + \gamma_P}$

```python
vdict[e_a] = eq_ea.rhs.subs(vdict)
vdict[e_d] = 0.5 * vdict[e_a]
vdict[f_u] = 0.1
expr1 = eq_16.rhs.subs(vdict)
expr2 = eq_8P.rhs.subs(vdict)
xvar = T_a
ylvar = LE
xmin = 273.
xmax = 320.
lefty = [(expr1, 'Penman'), (expr2, 'corr. PT'
plot_expr2(xvar, ylvar, xmin, xmax, lefty)
```

Create dictionary with forcing and computed data and progressively substitute into previously defined equations



Easily tracable and verifiable results!

Share entire notebook

renkulab.io/projects/stanislaus.schymanski/essm_egu_2020

No more guess-work!

Acknowledgements and Literature

We gratefully acknowledge support by the FNR ATTRACT programme (A16/SR/11254288).

Priestley, C. H. B. and Taylor, R. J.: On the Assessment of Surface Heat Flux and Evaporation Using Large-Scale Parameters, Month. Weath. Rev., 100(2), 81–92, 1972.

Penman, H. L.: Natural Evaporation from Open Water, Bare Soil and Grass, Proc. Roy. Soc. London. Series A, Mathematical and Physical Sciences, 193(1032), 120–145, 1948.

Luxembourg Institute of Science and Technology
5, avenue des Hauts-Fourneaux
L-4362 Esch/Alzette
phone: (+352) 275 888 - 1
fax: (+352) 275 885
stanislaus.schymanski@list.lu
LIST.lu