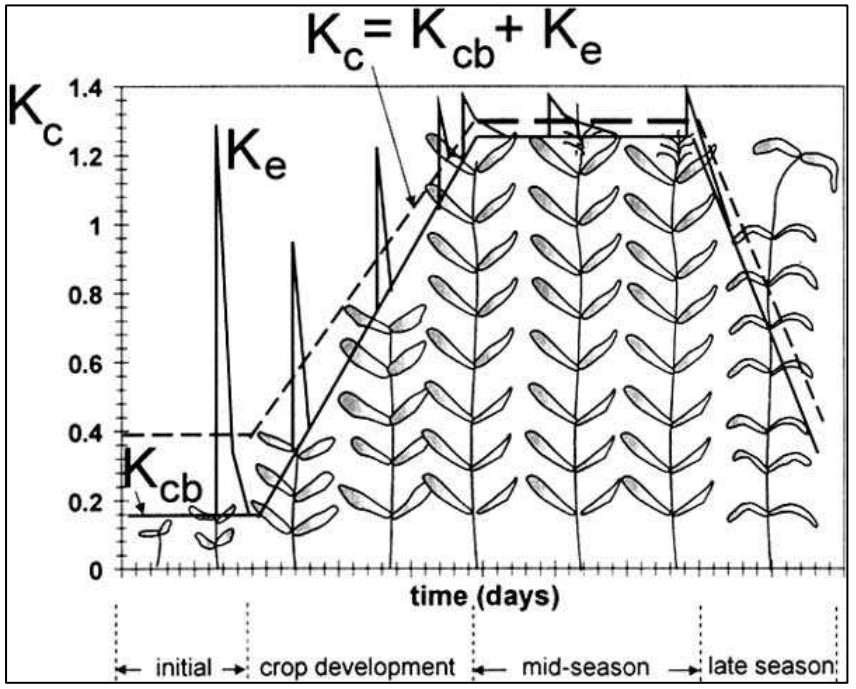
The “pyfao56” Package for Python: Codifying FAO-56 Evapotranspiration in a Modern Programming Language

Kelly R. Thorp, USDA-ARS, Temple, TX
Kendall DeJonge, Tyler Pokoski, Josh Brekel, Tom Trout, USDA-ARS, Fort Collins, CO
Meetpal Kukal, Dinesh Gulati, Penn State University
Ahmed Hashem, Fared Farag, Arkansas State University
Annelie Holzkaemper, Tamara Baumgartner, Gabriel Erismann, Agroscope, Zurich, Switzerland
and the pyfao56 development community

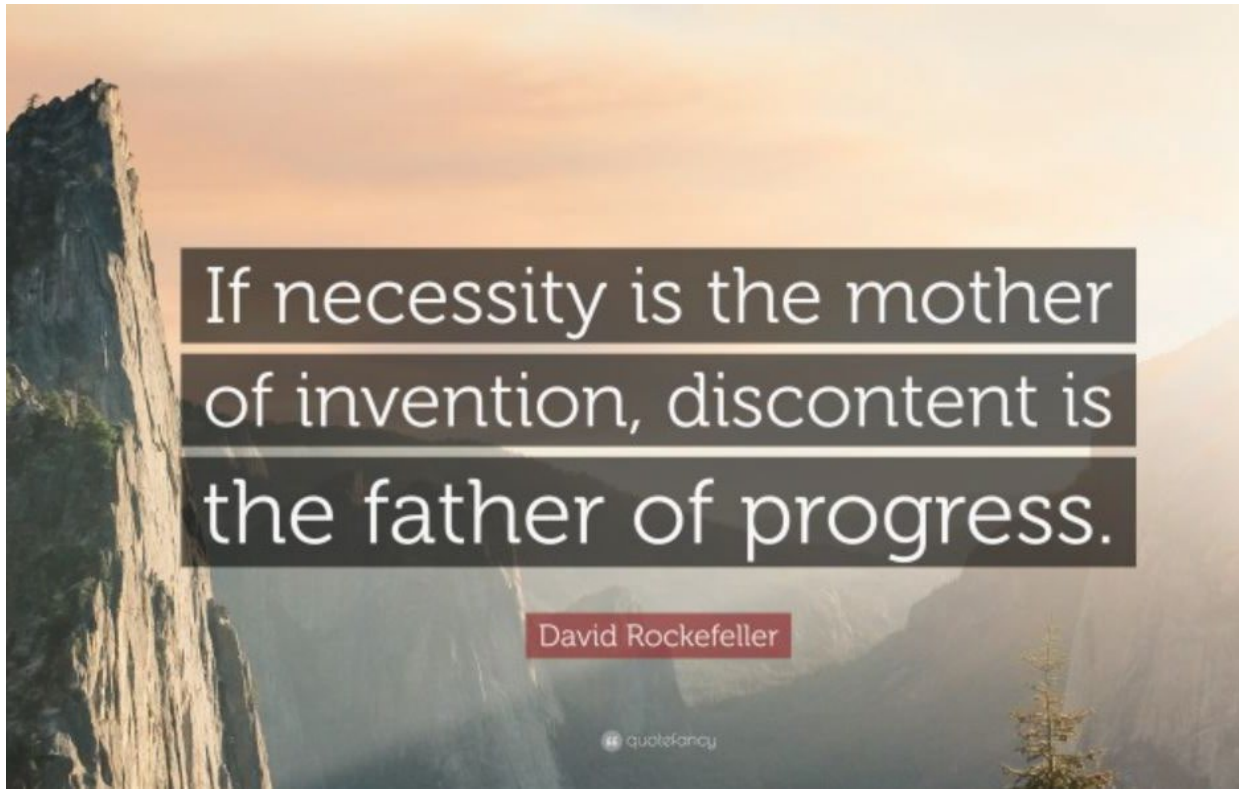
Introduction

- FAO-56 evapotranspiration
  - Allen et al. (1998) and ASCE (2005)
  - Standardized reference evapotranspiration (ET_o)
    - Defines atmospheric demand for water
    - Computed from weather station variables
    - air temp, dew point, solar radiation, and wind speed
  - Crop coefficients (K_c) adjust ET_o to crop ET (ET_c)
    - Single method: $ET_c = K_c \times ET_o$
    - Dual method: $ET_c = (K_{cb} \times K_s + K_e) \times ET_o$
- Python (www.python.org)
  - Highly popular modern programming language
  - Large development and user community
  - Open-source, easy syntax, lots of packages, etc.
- Goal: Codify FAO-56 algorithms in Python



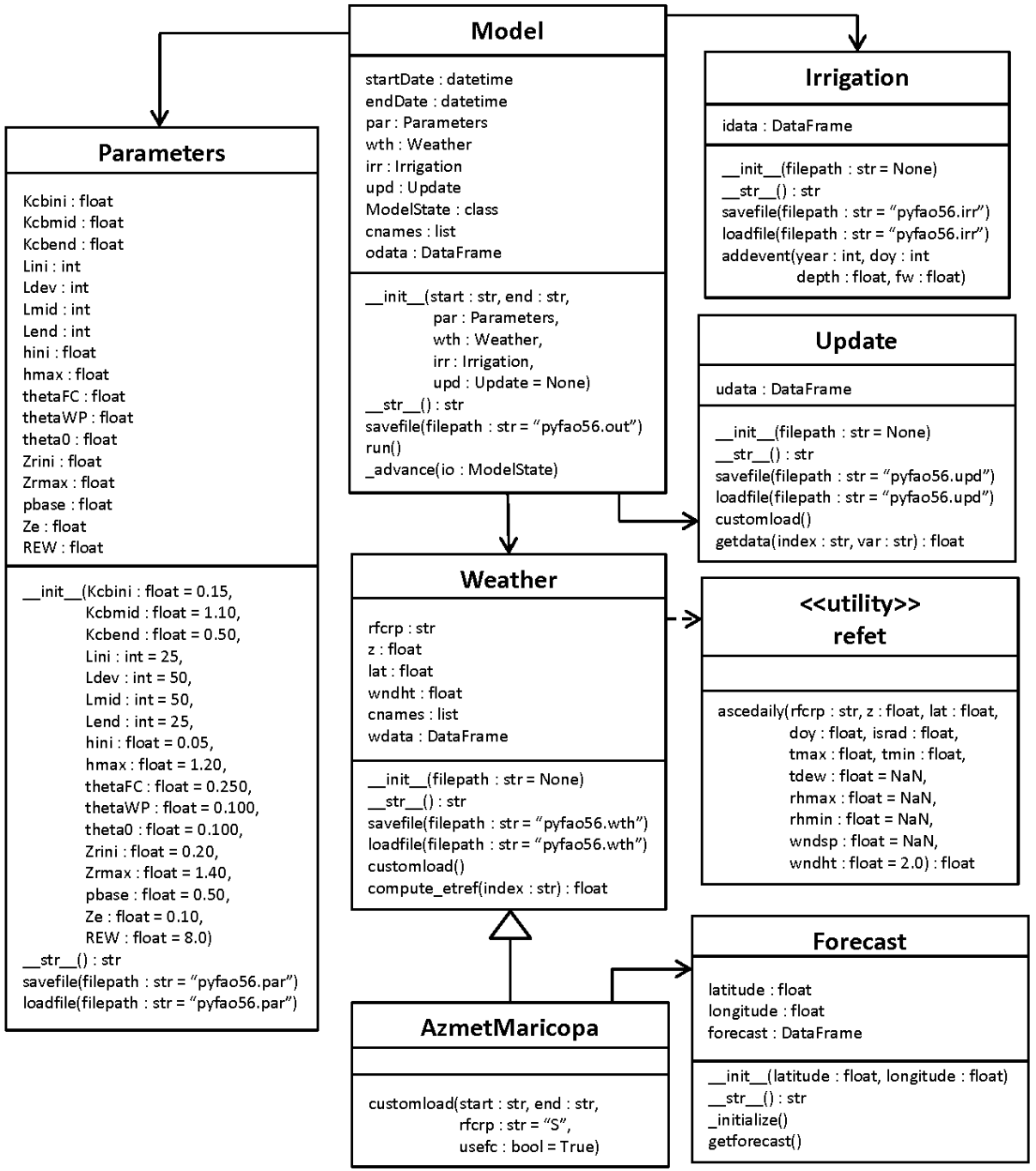
Software Design Goals

- Develop a Python tool for ET estimation and irrigation scheduling
  - Started as a script to support field studies
  - Then organized as a formal Python package
    - Importable to any Python module
    - Usable within other Python workflows
  - Eventually named it “pyfao56”
- Promote FAO-56 interpretation
  - Accurately represent FAO-56 methods
  - Clean, simple, readable, organized code
  - Well-documented and commented code
  - Serve as an extension of ET texts
  - Serve as a resource for learning FAO-56
- Promote accessibility of code
  - Open-source code
  - Distributed via Github, PyPI, conda-forge
  - Encourage collaborative development



Software Architecture

- Object-oriented modular design
  - Parameters
  - Irrigation
  - Autoirrigate (new in ver. 1.3.0)
  - Weather
  - SoilProfile (new in ver. 1.1.0)
  - Model
  - Update
  - Standardized reference ET module
- "pandas" DataFrames for data handling
- Multiple ways to inject data
  - Reading formatted text files
  - "customload" functions
    - Users can develop customized wrappers
    - Based on class inheritance
- Test suite and documented examples



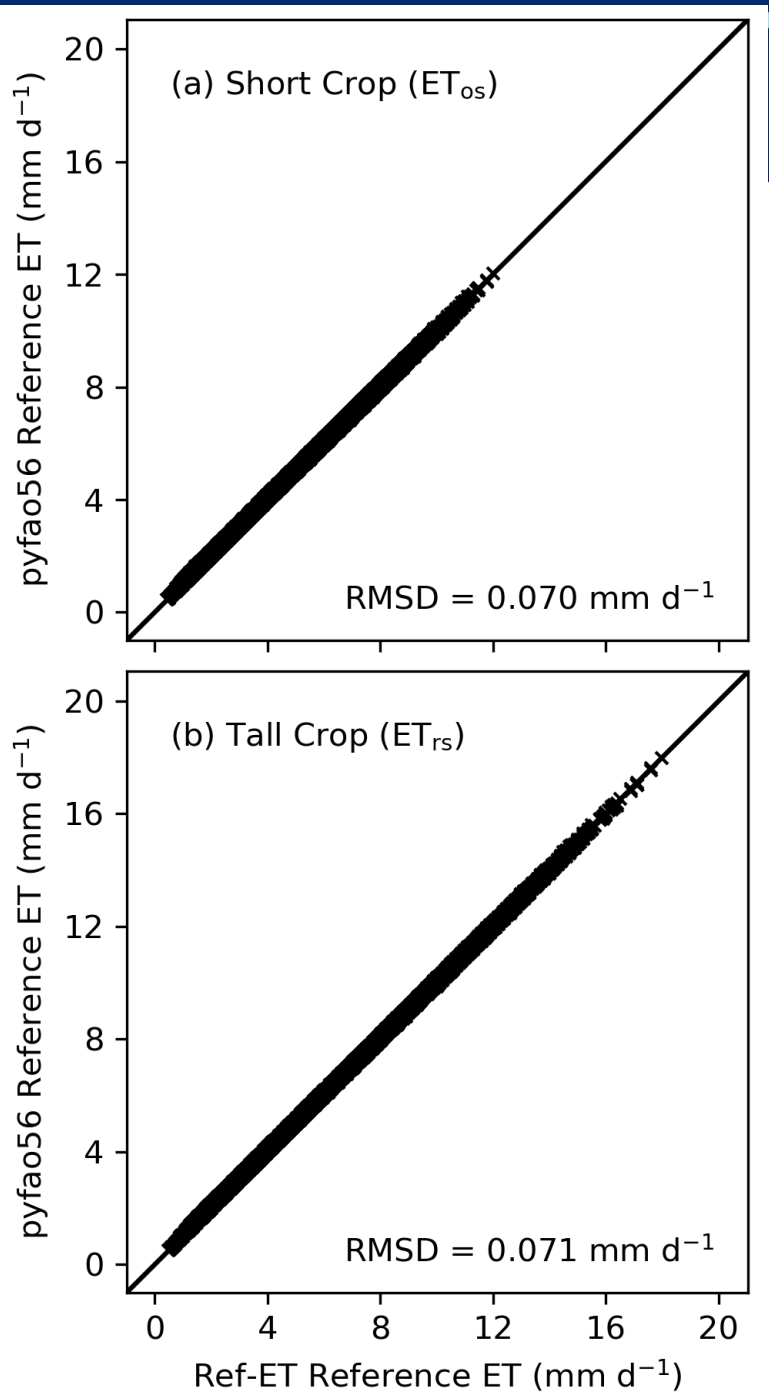
Quickstart

- Import pyfao56
- Specify model parameters
- Specify weather data
- Specify irrigation data
- Run the model
- More examples available

```
par.Kcbend = 0.15
par.Lini = 20
par.Ldev = 50
par.Lmid = 60
par.Lend = 30
par.hmax = 0.7
par.thetaFC = 0.2050
par.thetaWP = 0.0980
par.theta0  = 0.1919
par.Zrini = 0.20
par.Zrmax = 1.20
par.savefile(os.path.join(module_dir,'barley2023.par'))

#Specify the weather data
wth = custom.AzmetMaricopa()
wth.customload('2023-024','2023-121')
wth.savefile(os.path.join(module_dir,'barley2023.wth'))

#Specify the irrigation schedule
irr = fao.Irrigation()
irr.addevent(2023,  25, 20.4, 1.0)
irr.addevent(2023,  32, 20.4, 1.0)
irr.addevent(2023,  39, 20.4, 1.0)
irr.addevent(2023,  46, 10.2, 1.0)
irr.addevent(2023,  55, 20.4, 1.0)
irr.addevent(2023,  66, 20.4, 1.0)

#Run the model
mdl = fao.Model('2023-024','2023-121', par, wth, irr)
mdl.run()
print(mdl)
mdl.savefile(os.path.join(module_dir,'barley2023.out'))
```

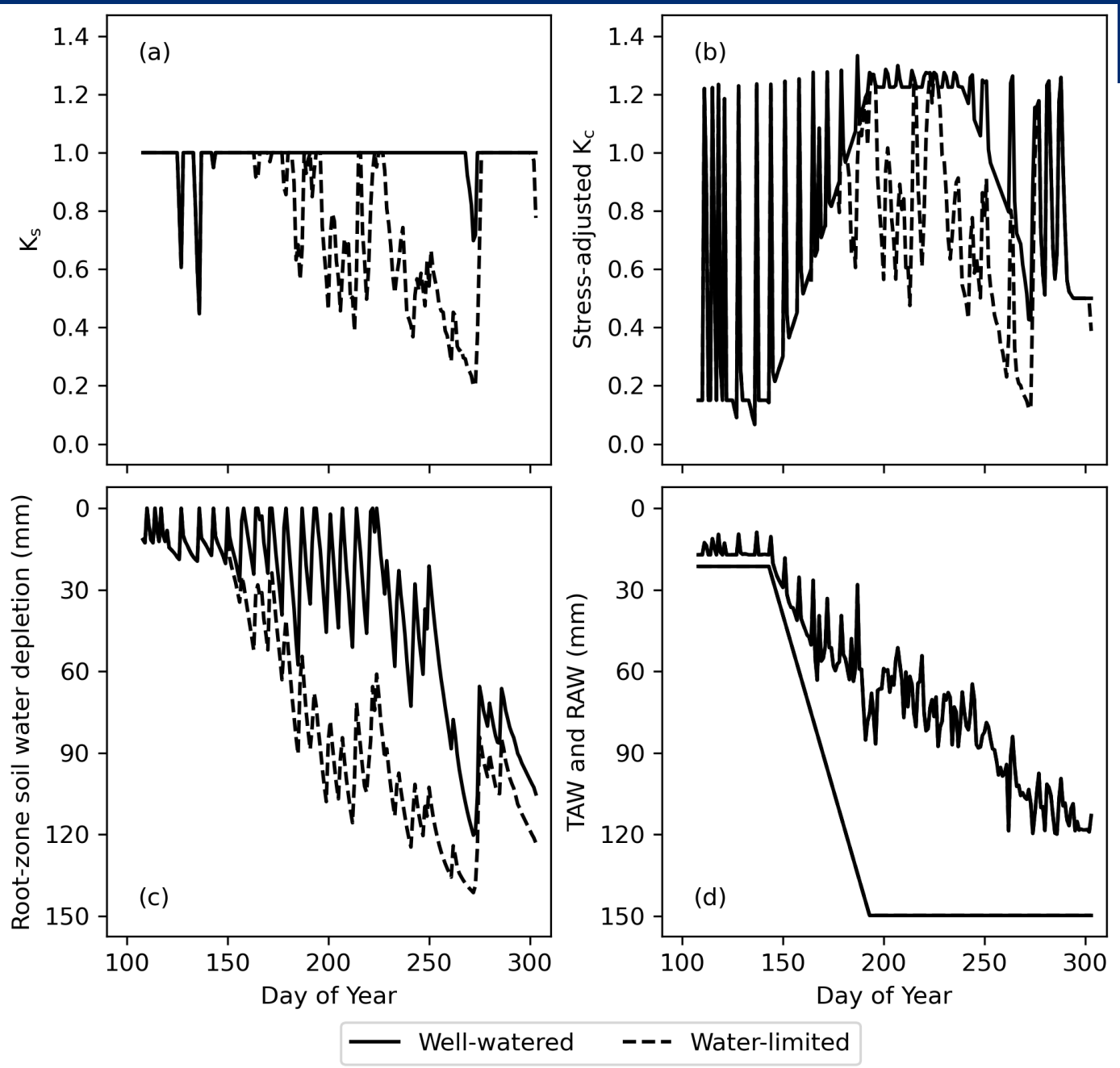
Reference Evapotranspiration

- How accurate are pyfao56 ET_{ref} computations?
  - No known open-source ET_{ref} algorithms
  - Can't borrow syntax from other sources
  - Can only follow the text in ASCE (2005)
- Compare ET_{ref} output from pyfao56 and Ref-ET software
  - Ref-ET software (Allen, 2012)
  - Known authority on ET_{ref} computation, but closed source
- Daily ET_{ref} comparison (pyfao56 vs. Ref-ET)
  - 18 years of AZMET weather data (2003 to 2020) at Maricopa, AZ
  - Short Crop ET_{ref} : RMSD = 0.070 mm d^{-1}
  - Tall Crop ET_{ref} : RMSD = 0.071 mm d^{-1}
- Hourly ET_{ref} comparison (pyfao56 vs. Ref-ET)
  - 5 years of weather data (2017 to 2021) at Greeley, CO
  - Short Crop ET_{ref}
    - RMSD (hourly data) = 0.007 mm h^{-1}
    - RMSD (daily sums) = 0.05 mm d^{-1}
  - Tall Crop ET_{ref}
    - RMSD (hourly data) = 0.011 mm h^{-1}
    - RMSD (daily sums) = 0.08 mm d^{-1}



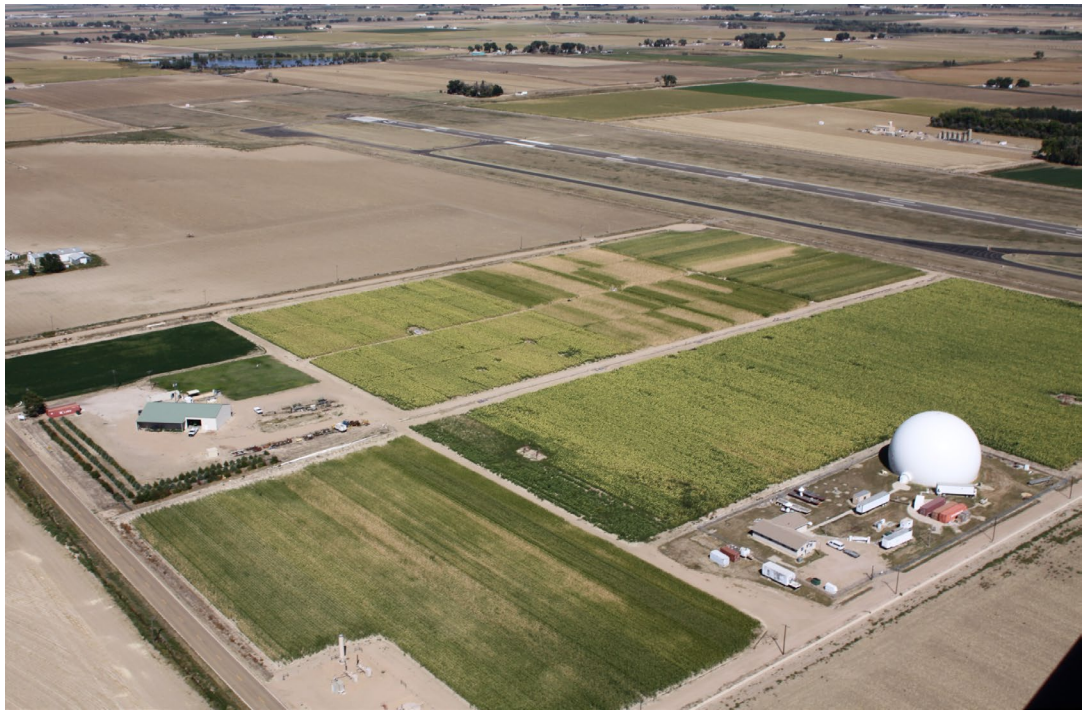
Model Response

- Model output
  - Fully represents all model states
  - Daily timestep data
  - Between start and end dates
- 2018 Maricopa cotton trial
  - Well-watered vs. water-limited
  - K_s responds to stress
  - D_r responds to stress
  - K_c spikes due to wetting events
  - TAW increases with root depth
  - Compare seasonal $ET_{c\ adj}$
    - pyfao56 vs. neutron water balance
    - Water-limited: +5.3% error
    - Well-watered: -5.6% error



Releases of pyfao56

- Initial development (ver. 1.0.9) by Thorp at Maricopa
  - Little input beyond the local research station
  - Favored specific FAO-56 methods used at Maricopa
- Later involvement with Fort Collins ARS
  - Incorporate their nuanced (better) FAO-56 implementations
  - But retain functionality of original software design
- Improvements from Fort Collins (ver. 1.1.0)
  - Inclusion of hourly ET_{ref} methodology
  - Consideration of stratified soil layers
    - FAO-56 uses only a single, homogeneous soil layer
    - Devised method for FAO-56 calculations with layered soil data
  - Enhanced soil water depletion calculations
    - FAO-56 does not account for water below dynamic root zone
    - Devised method to consider both dynamic and maximum root zones
  - Constant depletion fraction (p)
    - RAW = pTAW, but FAO-56 not clear on method for p computation
    - Devised method for considering p as constant or dynamic variable



ARS Limited Irrigation Research Farm (LIRF)
Greeley, Colorado

Releases of pyfao56

- Addition of “tools” subpackage (ver. 1.2.0)
  - New SoilWaterSeries and SoilWaterProfile classes
    - Integration of measured soil water content data
    - Computes measured root zone soil water depletion (D_r)
  - New Visualization class
    - Time series plots of depletion, crop coefficients, and ET
    - Comparisons of measured and modeled depletion
  - Updates to the Forecast class
- Growing community interest and user base (ver. 1.3.0)
  - Added runoff option to water balance (FAO-56 assumes runoff = 0)
  - Added comprehensive autoirrigation algorithm
  - Added considerations for irrigation efficiency
- Future plans
  - Soil water depletion data assimilation via direct insertion
  - Growth stage lengths based on growing degree days
  - Incorporate 2024 FAO56 revisions

More information

- Source code
  - Github: https://github.com/kthorp/pyfao56
  - Install to Python from Python Package Index (PyPI)
  - Install to conda from conda-forge
- SoftwareX articles
  - Thorp, K. R., 2022. pyfao56: FAO-56 evapotranspiration in Python. SoftwareX 19, 101208. doi:10.1016/softx.2022.101208
  - Brekel, J., Thorp, K. R., DeJonge, K. C., Trout, T. J., 2023. Version 1.1.0 – pyfao56: FAO-56 evapotranspiration in Python. SoftwareX 22, 101336. doi:10.1016/jsoftx.2023.101336
  - Thorp, K. R., Brekel, J., DeJonge, K. C., 2023. Version 1.2.0 – pyfao56: FAO-56 evapotranspiration in Python. SoftwareX 24, 101518. doi:10.1016/j.softx.2023.101518
  - Thorp, K. R., DeJonge, K. C., Pokoski, T., Gulati, D., Kukal, M., Farag, F., Hashem, A., Erismann, G., Baumgartner, T., Holzkaemper, A., 2024. Version 1.3.0 – pyfao56: FAO-56 evapotranspiration in Python. SoftwareX 26, 101724. doi:10.1016/j.softx.2024.101724

Thank you for your attention!

Kelly R. Thorp
kelly.thorp@usda.gov