The Journal of Open Source Software

MetSim: A Python package for estimation and disaggregation of meteorological data

Andrew R. Bennett[1], **Joseph J. Hamman**[2], **and Bart Nijssen**[1]

1 Department of Civil and Environmental Engineering, University of Washington **2** Climate and Global Dynamics Laboratory, National Center for Atmospheric Research

DOI: 10.21105/joss.02042

Software

- Review
- Repository
- Archive

Editor: Stefan Pfenninger

Reviewers:

- @Chilipp
- @dsryberg

Submitted: 16 January 2020
Published: 27 January 2020

Summary

Hydrometeorological modeling is concerned with regions that have uncertain or unknown boundary conditions. For example, incoming shortwave radiation, longwave radiation, and humidity are often observed with varying record lengths and observation intervals. Further, even when such quantities are measured it is often at a daily resolution, while many environmental models require finer temporal resolution for simulation. To provide the necessary data to solve the model equations in such circumstances we must be able to provide estimates for these quantities at the appropriate temporal resolution. `MetSim` is a Python package and standalone tool for the estimation of meteorological quantities at variable temporal resolutions that can address the issues described above. `MetSim` can be used to generate spatially distributed sub-daily timeseries of incoming shortwave radiation, outgoing longwave radiation, air pressure, specific humidity, relative humidity, vapor pressure, precipitation, and air temperature given daily timeseries of minimum temperature, maximum temperature, and precipitation. Figure 1 shows an example of `MetSim`'s transformations of these daily values into some of the available subdaily outputs. A summary of the available output variables and tunable parameters are included in the documentation.

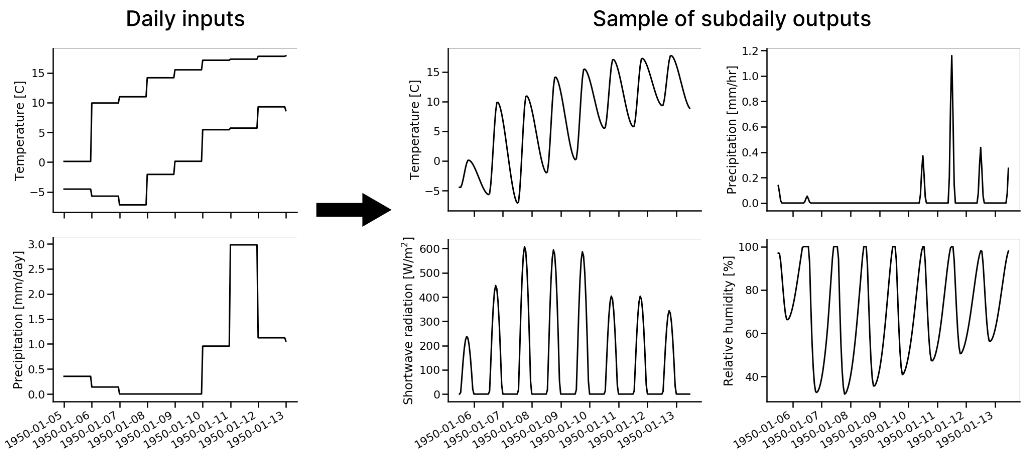

Figure 1: Figure 1: An example of `MetSim` input and output. The daily values are shown on the left are used as input along with a number of parameters to produce the hourly output shown on the right. The sample data used to generate this figure is included in the `MetSim` repository.

We have based `MetSim` on methods from the Mountain Microclimate Simulator (`MTCLIM`) and the forcing preprocessor that was built into the Variable Infiltration Capacity (`VIC`) hydrological model version 4 (Bohn et al., 2013; Liang, Lettenmaier, Wood, & Burges, 1994; Thornton

& Running, 1999). `MetSim` provides a modern workflow, building upon previous tools by improving performance, adding new IO routines, allowing for exact restarts, and providing an extensible architecture which can incorporate new features. We have designed `MetSim` to fit into the broader scientific Python ecosystem, building on popular packages such as `xarray` (Hoyer & Hamman, 2017), `dask`(Rocklin, 2015), `pandas`(McKinney, 2010), and `numba` (Lam, Pitrou, & Seibert, 2015).

Architecture and performance

`MetSim`'s architecture follows the common design of a model driver which coordinates high level operations and which delegates computation to several modules to do the bulk of the work. The top level model driver handles all IO routines as well as job scheduling and parallelism. A schematic representation of `MetSim`'s architecture is shown in figure 2. `MetSim` provides both a command line tool and API access as a Python module. The command line tool provides an interface to the driver via configuration files and command line options.

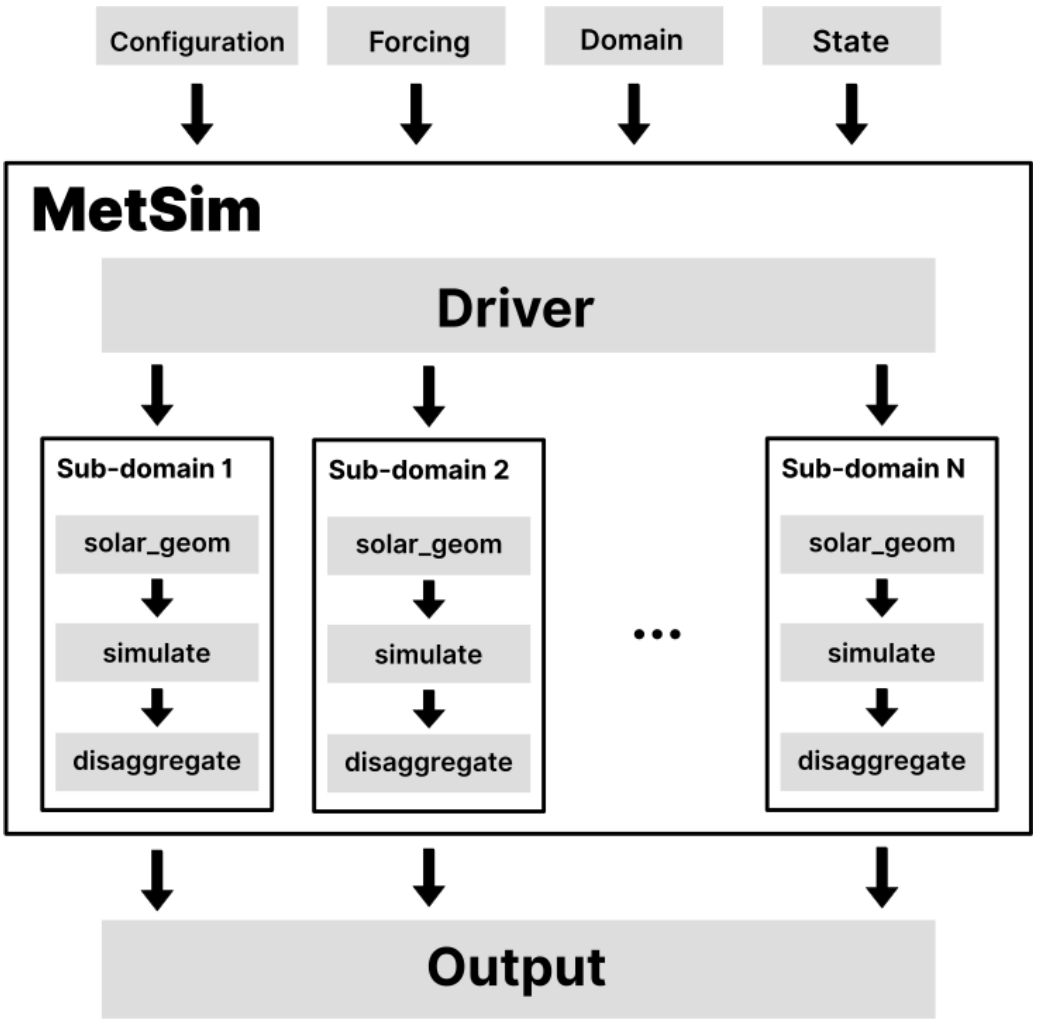

Figure 2: Figure 2: A schematic representation of the `MetSim` software flow

`MetSim` has three main computation modules for solar geometry, meteorological simulation, and temporal disaggregation. The solar geometry module computes the daily potential radiation, daylength, transmittance of the atmosphere, and the fraction of daily radiation received at the top of atmosphere during each 30 second interval. Computations are based on the algorithms described in Whiteman & Allwine (1986) as implemented in MTCLIM (Thornton & Running, 1999). The data from the solar geometry module is fed to the meteorology simulation module along with the input forcings. `MetSim` implements the estimation methods discussed in Bohn et al. (2013) and Thornton & Running (1999) to estimate the daily mean temperature, shortwave radiation, vapor pressure, and potential evapotranspiration. If disaggregation to shorter time steps is configured, the data is passed from the meteorology simulation module to the disaggregation module. Bohn et al. (2013) provides a further description and evaluation of these algorithms.

`MetSim` implements several options for parallelism, which are primarily managed by the `Dask` (Rocklin, 2015) and `xarray` (Hoyer & Hamman, 2017) libraries. We explore `MetSim`'s computational performance by conducting two scaling experiments. Strong scaling experiments test how the total runtime is affected by adding processors for a fixed overall problem size. Weak scaling experiments test how the total runtime is affected by adding processors proportional to the overall problem size. In ideal cases the runtime halves when doubling the number of processors for strong scaling experiments and remains constant for weak scaling experiments. These ideal cases are represented by the "perfect" cases shown in Figure 3. All of the times reported for the scaling experiments were for a single year run at an hourly time step with default parameter and output settings. For the strong scaling experiment we ran `MetSim` for one year at an hourly timestep over a domain of 6333 cells and ran using 2, 4, 8, 16, 32, 64, and 128 processors. The time to complete each run is shown in figure 3 panel a. The results show that scaling is nearly linear with the number of processors.

In the weak scaling experiment we ran `MetSim` using 2, 4, 8, 16, 32, 64, and 128 processors while varying the number of cells in the run to maintain a constant workload per processor. We ran 125 cells per 2 processors, resulting in runs of 125, 250, 500, 1000, 2000, 4000, and 8000 cells, respectively. The results of the weak scaling experiment are shown in panel b of figure 3. Similarly to the strong scaling experiment, we see increasing penalties for adding additional processors.

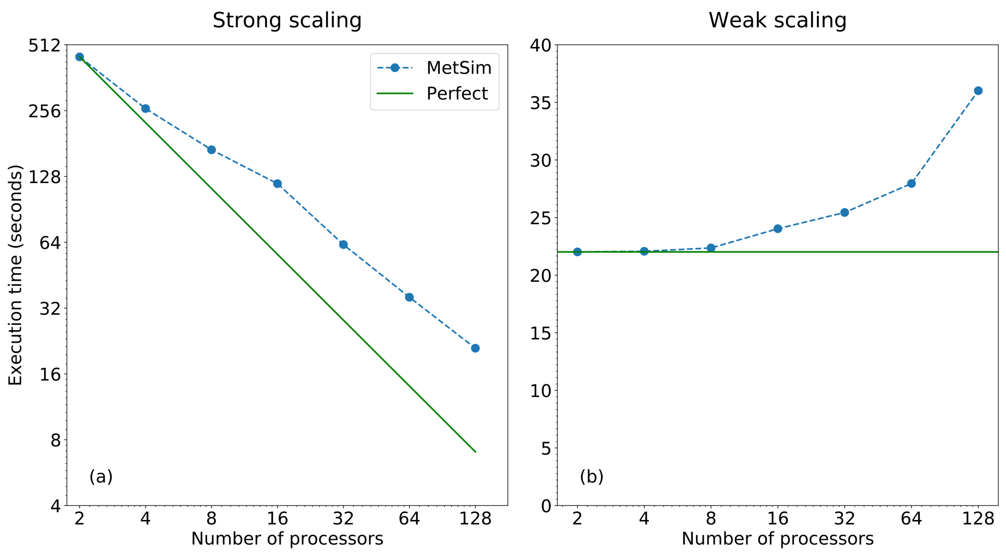

Figure 3: Figure 3: `MetSim` scaling performance

Applications & Related work

`MetSim` has been used in several research applications predominantly for generating input to hydrologic models, though other applications are possible. Bohn, Whitney, Mascaro, & Vivoni (2019) extended the precipitation disaggregation component to include a new option which was shown to result in better streamflow predictions than the default method. Cheng, Voisin, Yearsley, & Nijssen (2020) used `MetSim` as a component of their modeling framework to explore how reservoirs affect stream temperatures, and how reservoir operations may be able to help mitigate the effects of climate change on warming stream temperatures. The Climate Toolbox (Hegewisch, Abatzoglou, Chegwidden, & Nijssen, 2020) uses `MetSim` to generate meteorological data as an intermediate step for developing hydrologic predictions. `MetSim` has many other possible uses and is developer-friendly enough for it to be extended to provide additional functionality.

References

Bohn, T. J., Livneh, B., Oyler, J. W., Running, S. W., Nijssen, B., & Lettenmaier, D. P. (2013). Global evaluation of mtclim and related algorithms for forcing of ecological and hydrological models. *Agricultural and Forest Meteorology*, *176*, 38–49. doi:10.1016/j.agrformet.2013.03.003

Bohn, T. J., Whitney, K. M., Mascaro, G., & Vivoni, E. R. (2019). A deterministic approach for approximating the diurnal cycle of precipitation for use in large-scale hydrological modeling. *Journal of Hydrometeorology*, *0*(0), null. doi:10.1175/JHM-D-18-0203.1

Cheng, Y., Voisin, N., Yearsley, J., & Nijssen, B. (2020). Reservoirs modify river thermal regime sensitivity to climate change: A case study in the southeastern united states. *Water Resources Research, In Review*.

Hegewisch, K., Abatzoglou, J., Chegwidden, O., & Nijssen, B. (2020). Climate toolbox. Retrieved 2020, from https://climatetoolbox.org/

Hoyer, S., & Hamman, J. (2017). Xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, *5*(1). doi:10.5334/jors.148

Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A llvm-based python jit compiler. In *Proceedings of the second workshop on the llvm compiler infrastructure in hpc*, LLVM '15. New York, NY, USA: Association for Computing Machinery. doi:10.1145/2833157.2833162

Liang, X., Lettenmaier, D. P., Wood, E. F., & Burges, S. J. (1994). A simple hydrologically based model of land surface water and energy fluxes for general circulation models. *Journal of Geophysical Research: Atmospheres*, *99*(D7), 14415–14428. doi:10.1029/94JD00483

McKinney, W. (2010). Data structures for statistical computing in python. In S. van der Walt & J. Millman (Eds.), *Proceedings of the 9th python in science conference* (pp. 51–56).

Rocklin, M. (2015). Dask: Parallel computation with blocked algorithms and task scheduling. In K. Huff & J. Bergstra (Eds.), *Proceedings of the 14th python in science conference* (pp. 130–136).

Thornton, P. E., & Running, S. W. (1999). An improved algorithm for estimating incident daily solar radiation from measurements of temperature, humidity, and precipitation. *Agricultural and Forest Meteorology*, *93*(4), 211–228. doi:10.1016/S0168-1923(98)00126-9

Whiteman, C., & Allwine, K. (1986). Extraterrestrial solar radiation on inclined surfaces. *Environmental Software*, *1*(3), 164–169. doi:10.1016/0266-9838(86)90020-1