




# ronswanson: Building Table Models for 3ML

J. Michael Burgess <sup>1</sup>

<sup>1</sup> Max Planck Institute for Extraterrestrial Physics, Giessenbachstrasse, 85748 Garching, Germany

DOI: [10.21105/joss.04969](https://doi.org/10.21105/joss.04969)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

---

Editor: [Dan Foreman-Mackey](#) 

## Reviewers:

- [@cosimoNigro](#)
- [@volodymyrss](#)

Submitted: 14 October 2022

Published: 28 March 2023

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](https://creativecommons.org/licenses/by/4.0/)).

## Summary

ronswanson provides a simple-to-use framework for building so-called table or template models for `astromodels` ([Vianello et al., 2021](#)) the modeling package for multi-messenger astrophysical data-analysis framework, `3ML` ([Vianello et al., 2015-07](#)). With `astromodels` and `3ML` one can build the interpolation table of a physical model result of an expensive computer simulation. This then enables efficient reevaluation of the model while, for example, fitting it to a dataset. While `3ML` and `astromodels` provide factories for building table models, the construction of pipelines for models that must be run on high-performance computing (HPC) systems can be cumbersome. `ronswanson` removes this complexity with a simple, reproducible templating system. Users can easily prototype their pipeline on multi-core workstations and then switch to a multi-node HPC system. `ronswanson` automatically generates the required Python and SLURM scripts to scale the execution of `3ML` with `astromodels`'s table models on an HPC system.

## Statement of need

Spatio-spectral fitting of astrophysical data typically requires iterative evaluation of a complex physical model obtained from a computationally expensive simulation. In these situations, the evaluation of the likelihood is computationally intractable even on HPC systems. To circumvent this issue, one can create a so-called template or table model by evaluating the simulation on a grid of parameter values, then interpolating this output. Several spectral fitting packages, including `XSPEC` ([Arnaud, 1996](#)), `3ML`, and `gammapy` ([Acero et al., 2023](#); [Deil et al., 2017](#)), implement frameworks that allow for the reading these template models in various file formats. However, none of these libraries provide tools for uniformly generating the data from which these templates are built. `ronswanson` builds table models for `astromodels`, the modeling language of the multi-messenger data analysis framework `3ML` in an attempt to solve this problem. `astromodels` stores its table models as HDF5 ([The HDF Group, 1997-NNNN](#)) files. While `astromodels` provides a set of user-friendly factories for constructing table models, the workflow for using these factories on desktop workstations or HPC systems can be complex. However, these workflows are easily abstracted to a templating system that can be user-friendly and reproducible.

## Procedure

Once the user selects a simulation from which they would like to create a table model, the first task is to create an interface class that tells `ronswanson` how to run the simulation and collect its output. This is achieved by inheriting a class from the package called `Simulation` and defining its virtual `run` member function. With this function, the user specifies how the model parameters for each point in the simulation grid are fed to the simulation software. The outputs from the simulation are passed to a dictionary with a key for each different output. Finally, this dictionary is returned from the `run` function. This is all the programming that

is required as `ronswanson` uses this subclass to run the simulations on the user's specified architecture.

With the interface to the simulation defined, the user must specify the grid of parameters on which to compute the output of the simulation. This is achieved by specifying the grid points of each parameter in a YAML file. Parameter grids can either be custom, or specified with ranges and a specific number of evaluation points. Additionally, the energy grid corresponding to the evaluation of each of the simulation outputs must be specified in this file. The final step is to create a YAML configuration file telling `ronswanson` how to create the table. This includes specifying the name of the output HDF5 database, where to find the simulation subclass created in the first step, the name of the parameter YAML file, and details on the compute architecture on which the simulation grid is to be run.

With these two configuration files defined, the user runs the command line program `simulation_build` on the main configuration file. This automatically generates all the required Python and SLURM scripts required for the construction of the table model. If running on a workstation, the user then executes the `run_simulation.py` script. If, instead, the simulation is run on an HPC cluster, the user runs `sbatch run_simulation.sh`. In the case of running on an HPC system, the final step to build the database requires running `sbatch gather_results.sh` which uses MPI (Forum, 1994) to gather the individual pieces of the simulations into the main database.

The created HDF5 database can be loaded with utilities in `ronswanson` to then construct a table model in the `astromodels` format (see here for details). This intermediate step allows the user to select subsets of the parameters from which to construct the table model. This is useful as large interpolation tables can consume a lot of computer memory, and it is possible that certain fits may only need a limited parameter range. Additionally, utilities are provided that allow adding parameter sets onto the primary database to extend the interpolation range. Moreover, the database stores information such as the runtime of each grid point of the simulation. Utilities are provided to view this metadata. With future interfacing of 3ML and `gammapy`, these table models could even be used to fit data from optical to very high energy gamma-rays. More details and examples can be found in the [documentation](#).

## Acknowledgments

This project was inspired by earlier works of Elisa Schoesser and Francesco Berlato.

## References

- Acero, F., Aguasca-Cabot, A., Buchner, J., Carreto Fidalgo, D., Chen, A., Chromey, A., Contreras Gonzalez, J. L., Bony de Lavergne, M. de, Miranda Cardoso, J. V. de, Deil, C., Donath, A., Giunti, L., Hinton, J., Jouvin, L., Khélifi, B., King, J., Lefaucheur, J., Lenain, J.-P., Linhoff, M., ... Wood, M. (2023). *Gammapy: Python toolbox for gamma-ray astronomy* (Version v1.0.1). Zenodo. <https://doi.org/10.5281/zenodo.7734804>
- Arnaud, K. A. (1996). XSPEC: The First Ten Years. In G. H. Jacoby & J. Barnes (Eds.), *Astronomical data analysis software and systems v* (Vol. 101, p. 17).
- Deil, C., Zanin, R., Lefaucheur, J., Boisson, C., Khelifi, B., Terrier, R., Wood, M., Mohrmann, L., Chakraborty, N., Watson, J., Lopez-Coto, R., Klepser, S., Cerruti, M., Lenain, J. P., Acero, F., Djannati-Ataï, A., Pita, S., Bosnjak, Z., Trichard, C., ... Arribas, M. P. (2017). Gammapy - A prototype for the CTA science tools. *35th International Cosmic Ray Conference (ICRC2017)*, 301, 766. <https://doi.org/10.22323/1.301.0766>
- Forum, M. P. (1994). *MPI: A message-passing interface standard*. University of Tennessee.
- The HDF Group. (1997-NNNN1997-NNNN). *Hierarchical Data Format, version 5*.

Vianello, G., Burgess, J. M., Fleischhack, H., Di Lalla, N., & Omodei, N. (2021). *Astromodels* (Version 2.2.2). Zenodo. <https://doi.org/10.5281/zenodo.5646925>

Vianello, G., Lauer, R. J., Younk, P., Tibaldo, L., Burgess, J. M., Ayala, H., Harding, P., Hui, M., Omodei, N., & Zhou, H. (2015-07). *The Multi-Mission Maximum Likelihood framework (3ML)*. 1507, arXiv:1507.08343. <https://doi.org/10.22323/1.312.0130>