The Journal of Open Source Software

# FitBenchmarking: an open source Python package comparing data fitting software

**Anders Markvardsen[1], Tyrone Rees[1], Michael Wathen[1], Andrew Lister[1], Patrick Odagiu[1], Atijit Anuchitanukul[1], Tom Farmer[1], Anthony Lim[1], Federico Montesino[1], Tim Snow[2], and Andrew McCluskey[2]**

**1** Science and Technology Facilities Council, Rutherford Appleton Laboratory, Harwell Campus, Didcot, Oxfordshire, OX11 0QX **2** Diamond Light Source Ltd, Diamond House, Harwell Campus, Didcot, Oxfordshire, OX11 0DE

**DOI:** 10.21105/joss.03127

**Software**

- Review
- Repository
- Archive

**Editor:** David Hagan

**Reviewers:**

- @johnsamuelwrites
- @djmitche

**Submitted:** 01 March 2021
**Published:** 08 June 2021



## Summary

Fitting a mathematical model to data is a fundamental task across all scientific disciplines. FitBenchmarking has been designed to help:

- Scientists, who want to know the best algorithm for fitting their data to a given model using specific hardware.
- Scientific software developers, who want to identify the best fitting algorithms and implementations. This allows them to recommend a default solver, to see if it is worth adding a new minimizer, and to test their implementation.
- Mathematicians and numerical software developers, who want to understand the types of problems on which current algorithms do not perform well, and to have a route to expose newly developed methods to users.

Representatives of each of these communities have got together to build FitBenchmarking. We hope this tool will help foster fruitful interactions and collaborations across the disciplines.

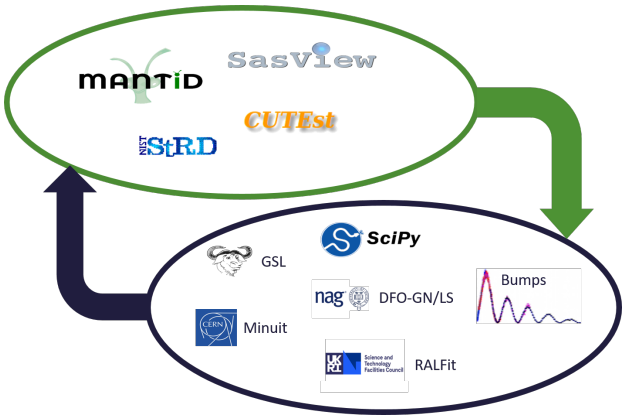


**Figure 1:** Benchmarking paradigm: associating fitting problems represented in individual scientific software packages (top cycle) to optimization software packages (bottom cycle), and bringing these closer together.

FitBenchmarking is easy to install via pip and our documentation guides users through the installation of some external packages we support. We provide several data sets from a range

of applications and adding new data in these formats is as easy as dropping the data into a new folder. The data and fitting packages currently supported are shown in Figure 1. A key part of `FitBenchmarking` is the ease of which a user, with a basic knowledge of `Python`, can add new fitting software, data formats and different fitting comparison output metrics.

## State of the field

Fitting data to models is a form of optimization, and `CUTEst` (Gould et al., 2015), and its predecessors, has been the standard tool to benchmark optimization packages for some time. `CUTEst` can benchmark any problem written in a custom `SIF` format. However, only the hooks to run the same problem are provided, the user must provide their own data analysis. Tools such as `Paver` (Bussieck et al., 2014), part of the COIN-OR initiative, can be used alongside `CUTEst` (or other tools) for this purpose. The packages `Olympus` (Häse et al., 2021) and `Benchopt` (*BenchOpt 1.1.0*, 2021) have been recently developed as benchmarking and analysis frameworks for optimization problems. `Olympus` is designed for experiment planning and provides analytic benchmark problems, experimental datasets, and emulated datasets, but could be adapted to be applied to any optimization (or data-fitting) problem. `Benchopt`, on the other hand, is currently primarily used to benchmark data fitting using a range of cost functions. `Benchopt` ships with a limited number of example data sets, but it is well documented how to write new benchmarks with custom data and objective functions.

## Statement of need

While there is some overlap between `FitBenchmarking` and the rest of the field, what makes our software unique is:

- It is designed to interface directly to the source of data, be that a scientific software package or an academic data set. Our `parser` class can be extended to make it clear what a developer needs to do to get data into `FitBenchmarking`.
- While being easy to extend using new software, or new data from currently supported packages, `FitBechmarking` ships with open datasets that all can use for testing.
- FitBenchmarking tests implementations of algorithms, not just algorithms. A growing number of optimization packages that can be used for data fitting are supported, and it is straightforward to extend our `controller` class to add new software.
- `FitBenchmarking` performs its own data processing and analysis and, if needed, the output generated can be customized for new data sets and/or minimizers.

As far as we are aware, `FitBenchmarking` is the only package that is designed specifically to interface directly with optimization packages and individual scientific software packages to test different implementations of fitting algorithms. `FitBenchmarking` originally started as a tool to benchmark fitting algorithms in the data reduction package `Mantid` (Arnold et al., 2014), which is used to process neutron scattering and muon spectroscopy data. `FitBenchmarking` has since been significantly extended to take data and models from other real world applications and data analysis / modelling / treatment packages, such as `SasView` (Doucet et al., 2020) and `CUTEst` (Gould et al., 2015). It fits models to the data by using a range of data fitting and nonlinear optimization software packages, and present comparisons through a variety of different metrics. These include comparison tables and performance profile plots.

`FitBenchmarking` compares how different fitting algorithms perform for the same data, model and initial guess. The best parameters for the model are found by solving a nonlinear least-squares problem, which can either be solved using a dedicated optimisation software package

or using a fitting algorithm implementation within a scientific software package. Figure 2 displays a data set from `FitBenchmarking` where the crosses are the data points and the two curves are the fits found by two optimization algorithms implemented in `GSL` (Galassi et al., 2009). From Figure 2, it is clear that the solution given by lmsder is better. As the volume of data increases, and we do more and more scientific analysis algorithmically, it is increasingly important that we apply the best available algorithm for a given category of fitting problems. `FitBenchmarking` generates HTML output that makes it easy to compare minimizers on a given problem set.

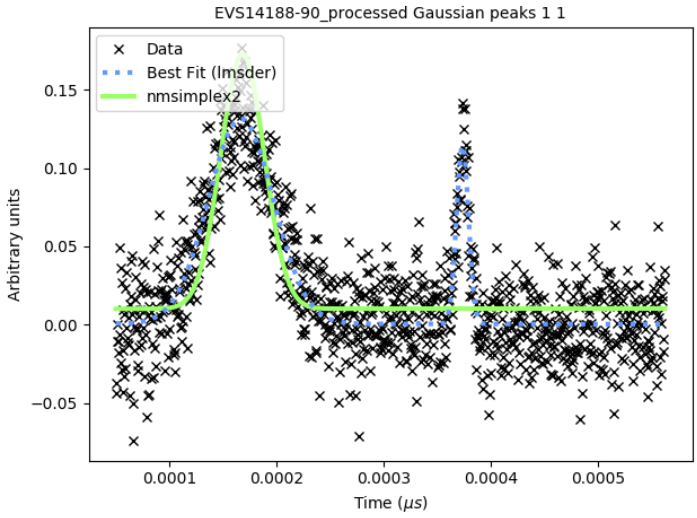


**Figure 2:** A sample fit: this problem is shipped with `FitBenchmarking`. The data was collected from an instrument named VESUVIO at the ISIS Neutron and Muon Source and has a difficult initial guess.

`FitBenchmarking` will help the scientist make an informed choice by comparing runtime and accuracy of all available minimizers, on their specific hardware, on problems from their science area.

`FitBenchmarking` will help the scientific software developer ensure that the most robust and quickest algorithms for the type of data analysis they support are available in their software.

`FitBenchmarking` will help mathematicians see what the state of the art is, and what kinds of data are problematic. It will give them access to real data, and will give a route for novel methods to quickly make it into production.

## Acknowledgements

We would like to acknowledge funding support from:

- European Union's Horizon2020 research and innovation programme, EU SINE2020 WP-10,
- EPSRC Grant EP/M025179/1 – Least Squares: Fit for the Future.
- The Ada Lovelace Centre (ALC).

We would also like to thank Nick Draper, Roman Tolchenov, Nick Gould and Jaroslav Fowkes for their helpful comments and advice.

## References


Arnold, O., Bilheux, J. C., Borreguero, J. M., Buts, A., Campbell, S. I., Chapon, L., Doucet, M., Draper, N., Ferraz Leal, R., Gigg, M. A., Lynch, V. E., Markvardsen, A., Mikkelson, D. J., Mikkelson, R. L., Miller, R., Palmen, K., Parker, P., Passos, G., Perring, T. G., … Zikovsky, J. (2014). Mantid—Data analysis and visualization package for neutron scattering and $\mu$ SR experiments. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, *764*, 156–166. https://doi.org/10.1016/j.nima.2014.07.029

*BenchOpt 1.1.0*. (2021). https://benchopt.github.io.

Bussieck, M. R., Dirkse, S. P., & Vigerske, S. (2014). PAVER 2.0: An open source environment for automated performance analysis of benchmarking data. *Journal of Global Optimization*, *59*(2-3), 259–275. https://doi.org/10.1007/s10898-013-0131-5

Doucet, M., Cho, J. H., Alina, G., Attala, Z., Bakker, J., Bouwman, W., Butler, P., Campbell, K., Cooper-Benun, T., Durniak, C., Forster, L., Gonzales, M., Heenan, R., Jackson, A., King, S., Kienzle, P., Krzywon, J., Nielsen, T., O'Driscoll, L., … Washington, A. (2020). *SasView version 5.0.3*. Zenodo. https://doi.org/10.5281/zenodo.3930098

Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G., Alken, P., Booth, M., & Rossi, F. (2009). *GNU scientific library reference manual* (B. Gough, Ed.; Third). Network Theory Ltd. ISBN: 0954612078

Gould, N. I., Orban, D., & Toint, P. L. (2015). CUTEst: A Constrained and Unconstrained Testing Environment with Safe Threads for Mathematical Optimization. *Comput. Optim. Appl.*, *60*(3), 545–557. https://doi.org/10.1007/s10589-014-9687-3

Häse, F., Aldeghi, M., Hickman, R. J., Roch, L. M., Christensen, M., Liles, E., Hein, J. E., & Aspuru-Guzik, A. (2021). *Olympus: A benchmarking framework for noisy optimization and experiment planning*. http://arxiv.org/abs/2010.04153