Bioinformatics, 2023, **39(11)**, btad711
https://doi.org/10.1093/bioinformatics/btad711
Advance Access Publication Date: 23 November 2023
Applications Note

Systems biology

pyPESTO: a modular and scalable tool for parameter estimation for dynamic models

Yannik Schälte [1,2,3,†], Fabian Fröhlich[4,†], Paul J. Jost[1,†], Jakob Vanhoefer[1,†], Dilan Pathirana[1], Paul Stapor[2,3], Polina Lakrisenko[2,5], Dantong Wang[2,3], Elba Raimúndez[1,2,3], Simon Merkt[1], Leonard Schmiester [2,3], Philipp Städter[2,3,6], Stephan Grein[1], Erika Dudkin[1], Domagoj Doresic[1], Daniel Weindl [2], Jan Hasenauer [1,2,3,*]

[1]Life and Medical Sciences (LIMES) Institute, University of Bonn, 53113 Bonn, Germany
[2]Computational Health Center, Helmholtz Zentrum München Deutsches Forschungszentrum für Gesundheit und Umwelt (GmbH), 85764 Neuherberg, Germany
[3]Department of Mathematics, Technical University of Munich, 85748 Garching, Germany
[4]Department of Systems Biology, Harvard Medical School, Boston, MA 02115, United States
[5]School of Life Sciences, Technical University of Munich, 85354 Freising, Germany
[6]Leibniz Institute for Natural Product Research and Infection Biology, 07745 Jena, Germany

*Corresponding author. Life and Medical Sciences (LIMES) Institute, University of Bonn, Endenicher Allee 64, 53113 Bonn, Germany. E-mail: jan.hasenauer@uni-bonn.de

†Equal contribution.

Associate Editor: Anthony Mathelier

Abstract

Summary: Mechanistic models are important tools to describe and understand biological processes. However, they typically rely on unknown parameters, the estimation of which can be challenging for large and complex systems. pyPESTO is a modular framework for systematic parameter estimation, with scalable algorithms for optimization and uncertainty quantification. While tailored to ordinary differential equation problems, pyPESTO is broadly applicable to black-box parameter estimation problems. Besides own implementations, it provides a unified interface to various popular simulation and inference methods.

Availability and implementation: pyPESTO is implemented in Python, open-source under a 3-Clause BSD license. Code and documentation are available on GitHub (https://github.com/icb-dcm/pypesto).

1 Introduction

In many research areas, including computational biology, mathematical models are important tools to study complex systems and understand underlying mechanisms (Kitano 2002). While there are a variety of formalisms to describe biological systems, ordinary differential equation (ODE) models are popular as they provide natural means to describe and explain dynamic changes after perturbations, common in experimental biology (Fröhlich *et al.* 2019). However, models usually have unknown parameters that need to be estimated—their value and uncertainty—from observed data (Tarantola 2005). We present pyPESTO, a Python-based parameter estimation tool that provides various inference approaches in a modular manner via a streamlined pipeline (Fig. 1; see the Supplementary Information, Section 1, for a tool comparison). pyPESTO originated as a freely accessible reimplementation of the MATLAB tool PESTO (Stapor *et al.* 2018), but now offers a pipeline with a much broader spectrum of modern methods and interfaced tools. See the Supplementary Information, Section 2, for an example workflow including problem definition, optimization and uncertainty quantification, demonstrating the usefulness of a streamlined pipeline and unified interface. pyPESTO can be applied to various problem types, including large-scale problems.

2 Features

2.1 Problem definition

To ease application to biological systems, pyPESTO supports the community standard PEtab. PEtab is an open-source interoperable format for the specification of parameter estimation problems in systems biology, covering model definition, data, parameters, and noise model (Schmiester *et al.* 2021). For simulation, pyPESTO interfaces in particular the ODE simulation and sensitivity engine AMICI (Fröhlich *et al.* 2021) and provides an interface to objective functions written in the Julia language, in particular DifferentialEquations.jl (Rackauckas and Nie 2017). Furthermore, pyPESTO allows for user-defined continuous parameter estimation problems given via scalar objective functions or vector-valued residual functions. This includes log-likelihood and log-posterior based objective functions, where estimated values and uncertainties can be interpreted statistically. As many inference methods benefit from objective function

Received: 3 May 2023; **Revised:** 2 October 2023; **Editorial Decision:** 13 November 2023; **Accepted:** 22 November 2023

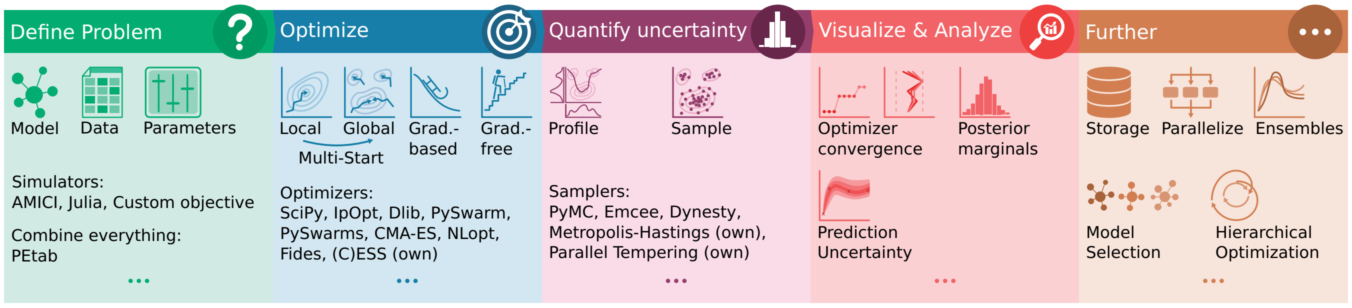

Figure 1. *Concept figure.* pyPESTO covers the full parameter estimation workflow, from problem definition (i.e. defining the parameter estimation problem based on dynamic model, data, parameters, and objective function), parameter optimization (i.e. finding parameters optimally explaining the data under the assumed objective function), uncertainty quantification (i.e. assessing uncertainty in parameter values), to visualization and analysis of results, and provides various features such as storage, parallelization, and advanced algorithms. See the main text for a contextualization of the key words indicated in the figure.

derivatives, pyPESTO supports user-supplied functions that compute derivatives. In particular, in combination with AMICI, gradients via forward and adjoint sensitivity analysis, which is substantially more scalable for large-scale problems, are available (Fröhlich *et al.* 2021). Further, pyPESTO provides central, backward and forward finite differences with adaptive step size selection.

2.2 Optimization

Finding globally optimal parameters as point estimates is a common starting point in parameter estimation. As many problems are nonconvex and possess multiple local optima, globalization strategies are necessary. To this end, pyPESTO provides interfaces to global optimizers as well as a multi-start globalization strategy for local and global optimizers, which performs well for biological problems (Raue *et al.* 2013). pyPESTO provides a unified interface to local and global optimization libraries such as Ipopt (Wächter and Biegler 2006), Dlib (King 2009), PySwarms (Miranda 2018), pycma (Hansen 2022), SciPy (Virtanen *et al.* 2020), NLopt (Johnson and Schueller 2021), and Fides (Fröhlich and Sorger 2022). An extension to other tools is easily possible. Moreover, pyPESTO provides a hierarchical approach to efficiently handle relative data and noise parameters (Schmiester *et al.* 2019) and ordinal data (Schmiester *et al.* 2020). Hierarchical optimization essentially decomposes the optimization problem into dynamic parameters (e.g. reaction rates) and static parameters (e.g. scaling factors and noise variables) that do not affect the dynamics but only the observation function. This decomposition results in an optimization problem that can be substantially easier to solve.

2.3 Uncertainty analysis

For uncertainty analysis, pyPESTO implements the optimization based (frequentist) profile likelihood and (Bayesian) profile posterior approaches using the aforementioned interfaces to different optimization libraries (Raue *et al.* 2009). Moreover, pyPESTO provides Bayesian sampling algorithms. It implements a basic Metropolis Markov-chain Monte Carlo algorithm with adaptive estimation of the correlation structure and acceptance rate based scaling (Miasojedow *et al.* 2013, Łącki and Miasojedow 2015, Ballnus *et al.* 2017), and a modular parallel tempering framework based on Vousden *et al.* (2016). Parallel tempering allows to traverse the posterior landscape at different "temperatures" and is highly suited for multi-modal problems. Further, pyPESTO provides a unified interface to the popular sampling tools Emcee (Foreman-Mackey *et al.* 2013), PyMC (Salvatier *et al.* 2016) with custom gradients, and Dynesty (Speagle 2020).

2.4 Further aspects

pyPESTO provides various routines to visualize and analyze obtained results. Results can be saved, recovered, and shared in a compact storage format based on HDF5. Moreover, pyPESTO supports shared-memory parallelization via multi-threading and multi-processing. Thus, pyPESTO can be deployed flexibly on desktop machines as well as high-performance-computing infrastructure.

2.5 Limitations

While broadly applicable and providing a streamlined pipeline for parameter estimation implementing and interfacing many inference algorithms, pyPESTO clearly also has some limitations. First, it focuses on parameter inference; for model simulation, it relies on tools such as AMICI and DifferentialEquations.jl. While any objective function and simulator can be used, some functionality of pyPESTO, such as model-data fit visualization and built-in PEtab support, are currently easiest to use in combination with ODE models simulated via AMICI. AMICI itself is a highly efficient C++-based ODE simulation and sensitivity calculation engine that supports a broad spectrum of ODE problems, including e.g. pre-equilibration, discrete events, parameterized likelihoods and different noise models; for details see Fröhlich *et al.* (2021). Further, while pyPESTO makes a wide range of inference methods and tools accessible, there are also some that it does not interface yet (e.g. currently Stan and JAX). Due to its modular interface, adding more tools is however straightforward when there is demand. Moreover, presently pyPESTO does not directly provide local or global sensitivity or identifiability methods, besides the provided uncertainty analysis methods. Last, in contrast to tools such as Copasi (Hoops *et al.* 2006), pyPESTO does not provide a graphical user interface (GUI) but is Python script-based, easiest to use e.g. with Jupyter notebooks.

3 Implementation and availability

pyPESTO is implemented in Python, open-source under a 3-Clause BSD license. The code, designed to be modular and extensible, is hosted on GitHub and can be installed from PyPI. Extensive documentation is hosted on ReadTheDocs, including numerous notebooks containing tutorials and outlining pyPESTO's functionality. "Getting started" tutorials allow to quickly get familiar with the basics of pyPESTO and a typical workflow including (custom) problem formulation, optimization, uncertainty quantification, visualization and analysis, and storage. Notebooks on "PEtab and AMICI" allow to quickly get familiar with workflows based on these tools. A further section on "Algorithms and features" provides detailed notebooks covering most of pyPESTO's components in depth. The "API documentation" provides a detailed description of the implemented functionality, including links to the code and to method details for further information. We ensure correctness during development via unit tests and continuous integration. pyPESTO is primarily targeted at Linux, which is of particular interest for high-performance computations, and is also being continually tested on MacOS and Windows.

4 Discussion

pyPESTO has already been used in several research projects and publications, and is continuously being developed as a community tool by core contributors from different institutions. In the future, we plan to implement additional optimization and uncertainty quantification algorithms, to interface pyPESTO with further popular tools, and to extend and further standardize the supported parameter estimation workflows. We anticipate that pyPESTO will continue to be useful in a variety of computational biology applications and beyond.

Supplementary data

Supplementary data are available at *Bioinformatics* online.

Conflict of interest

None declared.

Funding

This work was supported by the German Research Foundation (DFG) under Germany's Excellence Strategy [EXC 2047/1 390685813, EXC 2151 390873048]; the SFB 1454 [Metaflammation, 432325352]; TRR333 [BATenergy, 450149205]; AMICI [443187771]; the Human Frontier Science Program [LT000259/2019-L1 to F.F.]; the German Federal Ministry of Education and Research (BMBF) [01ZX1916A to D.D., D.W., and P.L.]; a Schlegel Professorship at the University of Bonn (to J.H.); and the Joachim Herz Foundation (to Y.S.).

References

Ballnus B, Hug S, Hatz K *et al.* Comprehensive benchmarking of Markov chain Monte Carlo methods for dynamical systems. *BMC Syst Biol* 2017;**11**:63.

Foreman-Mackey D, Hogg DW, Lang D *et al.* emcee: the MCMC hammer. *Publ Astronom Soc Pac* 2013;**125**:306–12.

Fröhlich F, Loos C, Hasenauer J. Scalable inference of ordinary differential equation models of biochemical processes. In: Sanguinetti G and Huynh-Thu VA (eds), *Gene Regulatory Networks: Methods and Protocols, Volume 1883 of Methods in Molecular Biology*, Chapter 16, 1st edn. New York, NY, USA: Humana Press, 2019, 385–422.

Fröhlich F, Sorger PK. Fides: reliable trust-region optimization for parameter estimation of ordinary differential equation models. *PLoS Comput Biol* 2022;**18**:e1010322.

Fröhlich F, Weindl D, Schälte Y *et al.* AMICI: high-performance sensitivity analysis for large ordinary differential equation models. *Bioinformatics* 2021;**37**:3676–7.

Hansen N. *CMA-ES/pycma: r3.2.2*. Zenodo 2022. https://doi.org/10.5281/zenodo.6370326.

Hoops S, Sahle S, Gauges R *et al.* COPASI – a COmplex PAthway SImulator. *Bioinformatics* 2006;**22**:3067–74.

Johnson SG, Schueller J. NLopt: nonlinear optimization library. In: *Astrophysics Source Code Library*. Houghton, MI, USA: ASCL, 2021, pp. ascl–2111.

King DE. Dlib-ml: a machine learning toolkit. *J Mach Learn Res* 2009;**10**:1755–8.

Kitano H. Systems biology: a brief overview. *Science* 2002;**295**:1662–4.

Łącki MK, Miasojedow B. State-dependent swap strategies and automatic reduction of number of temperatures in adaptive parallel tempering algorithm. *Stat Comput* 2015;**26**:951–64.

Miasojedow B, Moulines E, Vihola M. An adaptive parallel tempering algorithm. *J Comput Graph Stat* 2013;**22**:649–64.

Miranda LJV. PySwarms, a research-toolkit for particle swarm optimization in Python. *J Open Source Softw* 2018;**3**:433.

Rackauckas C, Nie Q. Differentialequations.jl—a performant and feature-rich ecosystem for solving differential equations in Julia. *JORS* 2017;**5**:15.

Raue A, Kreutz C, Maiwald T *et al.* Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics* 2009;**25**:1923–9.

Raue A, Schilling M, Bachmann J *et al.* Lessons learned from quantitative dynamical modeling in systems biology. *PLoS One* 2013;**8**:e74335.

Salvatier J, Wiecki TV, Fonnesbeck C *et al.* Probabilistic programming in Python using PyMC3. *PeerJ Comput Sci* 2016;**2**:e55.

Schmiester L, Schälte Y, Fröhlich F *et al.* Efficient parameterization of large-scale dynamic models based on relative measurements. *Bioinformatics* 2019;**36**:594–602.

Schmiester L, Weindl D, Hasenauer J. Statistical inference of mechanistic models from qualitative data using an efficient optimal scaling approach. *J Math Biol* 2020;**81**:603–23.

Schmiester L, Schälte Y, Bergmann FT *et al.* PEtab—interoperable specification of parameter estimation problems in systems biology. *PLoS Comput Biol* 2021;**17**:e1008646.

Speagle JS. Dynesty: a dynamic nested sampling package for estimating Bayesian posteriors and evidences. *Mon Not R Astron Soc* 2020;**493**:3132–58.

Stapor P, Weindl D, Ballnus B *et al.* PESTO: Parameter EStimation TOolbox. *Bioinformatics* 2018;**34**:705–7.

Tarantola A. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Philadelphia, Pennsylvania, USA: SIAM, 2005.

Virtanen P, Gommers R, Oliphant TE *et al.*; SciPy 1.0 Contributors. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat Methods* 2020;**17**:261–72.

Vousden WD, Farr WM, Mandel I *et al.* Dynamic temperature selection for parallel tempering in Markov chain Monte Carlo simulations. *Mon Not R Astron Soc* 2016;**455**:1919–37.

Wächter A, Biegler LT. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math Program* 2006;**106**:25–57.