The Journal of Open Source Software

LikelihoodProfiler.jl: Unified profile-likelihood workflows for identifiability and confidence intervals

Ivan Borisov [1], **Aleksander Demin**[2], **and Evgeny Metelkin** [1]

1 InSysBio CY LTD, Limassol, Cyprus **2** National Research University Higher School of Economics, Moscow, Russia

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 04 September 2025
Published: unpublished

Summary

Practical identifiability addresses how well a mechanistic model is determined by available experimental data. Profile likelihood-based methods are widely used to determine practical identifiability and serve as a proxy for structural identifiability analysis, particularly when the complexity of a model makes structural methods impractical (Heinrich et al., 2025). Profile likelihood techniques can be extended beyond parameter analysis to assess the identifiability of model states and predictions. This versatility makes profile likelihood analysis an essential component in the development and validation of models.

`LikelihoodProfiler.jl` is an open-source Julia package designed to perform profile likelihood-based identifiability analyses by offering a unified and extensible interface.

Statement of Need

Despite the widespread use of profile likelihood methods in practical identifiability analysis, existing tools often lack a common interface and extensive ecosystem integration, limiting accessibility and reproducibility. Also different profile likelihood based methods are implemented in different software tools, which requires the user to switch between languages and software interfaces. `LikelihoodProfiler.jl` addresses these limitations by providing: - Unified interface to access multiple profiling methods. - Compatibility with common modeling standards (Heta (Metelkin, 2021), PEtab (Persson & others, 2025), SBML). - Integration with Julia's SciML (Rackauckas & Nie, 2017) for efficient computation and extensibility.

Features and Methodologies

`LikelihoodProfiler.jl` provides a unified interface for various profile likelihood methods, including optimization-based (**OptimizationProfiler**) and integration-based profiles (**IntegrationProfiler**), CI endpoints search (**CICOProfiler**), and more.

All methods leverage a CommonSolve interface (Rackauckas & Nie, 2017) (`CommonSolve.solve()`), supporting global settings for parallelization and verbosity. Profiling results are directly visualizable using the `Plots.jl` package and exportable as DataFrames for further analysis.

Demonstrative Example: JAK/STAT Signaling Pathway Model

`LikelihoodProfiler.jl`'s functionality and interfaces are demonstrated using the JAK/STAT signaling pathway model (Boehm et al., 2014), which consists of 8 states and 9 parameters. The model and experimental data were sourced from the Benchmark-Models-PEtab repository (contributors, 2024) and imported through the `PEtab.jl` interface.

```
using PEtab, Plots
petab_model = PEtabModel("Boehm_JProteomeRes2014.yaml")
petab_problem = PEtabODEProblem(petab_model)
```

To define a profile likelihood problem `ProfileLikelihoodProblem` one should provide the objective function (usually negative log likelihood) and the initial (optimal) values of the parameters that correspond to the minimum of the objective function. `LikelihoodProfiler` relies on the `Optimization.jl` interface (Dixit & Rackauckas, 2023), and `ProfileLikelihoodProblem` is built on top of the `OptimizationProblem` defined in `Optimization.jl`. `ProfileLikelihoodProblem` also allows users to specify optional arguments common to different profiling methods, such as profile_range, threshold, and others:

```
using Optimization, LikelihoodProfiler
optprob = OptimizationProblem(petab_problem)
plprob = PLProblem(optprob, get_x(petab_problem))
```

`LikelihoodProfiler.jl` offers a suite of methods for profiling likelihood functions and assessing practical identifiability. Each method includes several configurable options, such as optimizer or integrator selection, tolerances, and step size.

The most straightforward method is `OptimizationProfiler`, which follows the classical approach of stepwise re-optimization of the likelihood function under a constraint on the parameter of interest. The method benefits from the choice of optimization algorithm (e.g., gradient-based or derivative-free) available throght `Optimization.jl` interface.

```
alg1 = OptimizationProfiler(optimizer = Optimization.LBFGS(), stepper = FixedStep(; init
```

A more advanced method is the `IntegrationProfiler`, which computes likelihood profiles by solving a system of differential equations derived from the underlying optimization problem. It provides smooth profile trajectories but requires Hessian computation or approximation. This method requires a differential equation solver (integrator) to be specified.

```
using OrdinaryDiffEq
alg2 = IntegrationProfiler(integrator = Tsit5(), integrator_opts = (dtmax=0.07,), matrix
```

An alternative approach, implemented in `CICOProfiler`, estimates the confidence intervals (CI) endpoints directly — without reconstructing the full profile — by solving a constrained optimization problem (Borisov & Metelkin, 2020). This method is often more efficient when only the CI is required.

```
alg3 = CICOProfiler(optimizer = :LN_NELDERMEAD, scan_tol = 1e-10)
```

All profiling methods share a common `solve()` interface.

```
sol = solve(plprob, alg1)
```

All three profiling approaches yielded comparable confidence intervals, emphasizing the reliability and flexibility of `LikelihoodProfiler.jl` for different modeling scenarios.

Below are the profile likelihoods for the first three parameters of the JAK/STAT model, computed using the three methods:

The Journal of Open Source Software

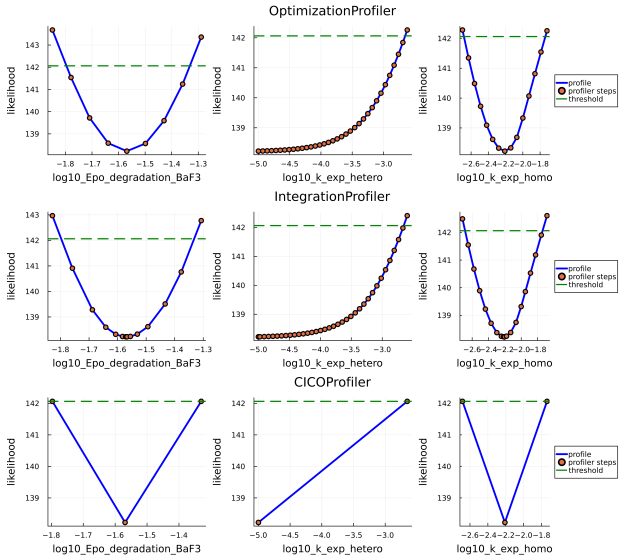

Figure 1: Profiles.

All three methods reported similar CI for the JAK/STAT model, which can be accessed using the `get_endpoint()` function.

The optimal profiling method and settings depend on the complexity of the model and the goal of the analysis.

Implementation and Extensibility

All profiling methods benefit from the unified interface provided by `LikelihoodProfiler.jl`: - Integration with `SciML` packages gives users access to a wide range of optimizers, differential equation solvers, and AD backends, enabling efficient profiling configurations. - Compatibility with `Heta`, `PEtab` and `SBML` formats broadens the accessibility of the package across different modeling frameworks. - `solve()` interface provided by `CommonSolve.jl` provides unified access various profiling methods - A common parallelization setup, controlled via the `parallel_type` argument in the `solve()` function, is supported across all methods and can significantly accelerate computations. - The interface facilitates integration of new profiling methods and stepping algorithms.

Future work will include adding new methods of parameters, functions and predictions profiling and enabling adaptive switching between strategies.

Availability

LikelihoodProfiler.jl is open-source and available at: https://github.com/insysbio/LikelihoodProfiler.jl

The package is registered in Julia and can be installed from the Julia REPL using:

```
import Pkg; Pkg.add("LikelihoodProfiler")
```

Tutorials and documentation are available at: https://insysbio.github.io/LikelihoodProfiler.jl/stable/

Related packages

Julia packages, such as `ProfileLikelihood.jl` and `InformationGeometry.jl`, offer related functionality but do not provide multiple profiling methods through a unified interface. Outside

Julia, tools like `Data2Dynamics (MATLAB)`, `dMod (R)`, and `pyPESTO (Python)` combine parameter estimation and profile likelihoods, but methods are often confined to different packages and languages. LikelihoodProfiler.jl brings multiple strategies together in a single, Julia-native API, reducing context-switching and making method selection a matter of configuration.

References

Boehm, M. E., Adlung, L., Schilling, M., Roth, S., Klingmüller, U., & Lehmann, W. D. (2014). Identification of isoform-specific dynamics in phosphorylation-dependent STAT5 dimerization by quantitative mass spectrometry and mathematical modeling. *J. Proteome Res.*, *13*(12), 5685–5694. https://doi.org/10.1021/pr5006923

Borisov, I., & Metelkin, E. (2020). Confidence intervals by constrained optimization—an algorithm and software package for practical identifiability analysis in systems biology. *PLoS Comput Biol*, *16*(12), e1008495. https://doi.org/10.1371/journal.pcbi.1008495

contributors, T. Pe. B. C. (2024). *The PEtab Benchmark Collection of parameter estimation problems* (Version 2024.10.15). Zenodo. https://doi.org/10.5281/zenodo.8155057

Dixit, V. K., & Rackauckas, C. (2023). *Optimization.jl: A unified optimization package* (Version v3.12.1). Zenodo. https://doi.org/10.5281/zenodo.7738525

Heinrich, M., Rosenblatt, M., Wieland, F.-G., Stigter, H., & Timmer, J. (2025). On structural and practical identifiability: Current status and update of results. *Current Opinion in Systems Biology*, *50*, 100546. https://doi.org/10.1016/j.coisb.2025.100546

Metelkin, E. (2021). Heta compiler: A software tool for the development of large-scale QSP models and compilation into simulation formats. *JOSS*, *6*(67), 3708. https://doi.org/10.21105/joss.03708

Persson, S., & others. (2025). *PEtab.jl: Advancing the efficiency and utility of dynamic modelling*. https://doi.org/10.1101/2025.04.30.651378

Rackauckas, C., & Nie, Q. (2017). DifferentialEquations.jl – a performant and feature-rich ecosystem for solving differential equations in julia. *JORS*, *5*(1), 15. https://doi.org/10.5334/jors.151