PEtab Select: specification standard and supporting software for automated model selection

Dilan Pathirana[1*], Frank T. Bergmann[4], Domagoj Doresic[1], Polina Lakrisenko[1], Sebastian Persson[5], Niklas Neubrand[6], Jens Timmer[7,8,9], Clemens Kreutz[6,8,9], Harald Binder[6,8,9], Marija Cvijovic[5], Daniel Weindl[1,2], Jan Hasenauer[1,2,3*], PEtab Select developer group

1 Life and Medical Sciences (LIMES) Institute & Bonn Center for Mathematical Life Sciences, University of Bonn, 53115 Bonn, Germany
2 Computational Health Center, Helmholtz Zentrum München Deutsches Forschungszentrum für Gesundheit und Umwelt (GmbH), 85764 Neuherberg, Germany
3 School of Life Sciences, Technische Universität München, Freising, Germany
4 BioQUANT, Heidelberg University, 69120 Heidelberg, Germany
5 Department of Mathematical Sciences, Chalmers University of Technology, and University of Gothenburg, Gothenburg, Sweden
6 Institute of Medical Biometry and Statistics (IMBI), Medical Faculty and Medical Center – University of Freiburg, Germany
7 Institute of Physics, University of Freiburg, Germany
8 CIBSS, Centre for Integrative Biological Signalling Studies – University of Freiburg, Germany
9 Freiburg Center for Data Analysis and Modelling – University of Freiburg, Germany

* dilan.pathirana@uni-bonn.de
* jan.hasenauer@uni-bonn.de

Abstract

A central question in mathematical modeling of biological systems is determining which processes are relevant and how they can be described. There are often competing hypotheses, which yield different models. Model comparison requires parameter optimization and sampling methods. Yet, standards for the specification of model selection problems and the swift evaluation of a broad spectrum of approaches are not available.

PEtab Select addresses this challenge by providing a concise, standardized specification of model selection and its associated calibration problems through a new file format standard and software package. It facilitates the compact representation of even very large model selection problems; in one example, billions of model alternatives. PEtab Select builds on the PEtab standard for the specification of parameter estimation problem and enables the use of state-of-the-art modelling and calibration workflows, i.e., COPASI, Data2Dynamics, PEtab.jl, and pyPESTO. It supports common model selection criteria (e.g., Akaike and Bayesian information criterion) and can be easily extended to use others. To ensure flexibility, PEtab Select implements several model space exploration approaches, including basic brute-force, forward, and backward selection, and also advanced, flexible selection methods.

PEtab Select introduces the first standardization of model selection tasks, filling a critical gap in existing computational pipelines. It constitutes an essential contribution to FAIR research software in systems biology by promoting interoperability and reusability in model selection.

Author summary

Model selection is a crucial step in mathematical modeling, guiding the choice of components to include in a model. PEtab Select facilitates this process across diverse modeling frameworks and programming languages via (1) a new interoperable, language-agnostic standard for specifying large-scale model selection problems, and (2) a comprehensive software package that implements these selection methods. Developed through a community effort, PEtab Select has been integrated into multiple modeling frameworks and is accessible to users of COPASI, Julia, MATLAB, Python.

Introduction

In systems biology, mechanistic models are employed for various purposes, including explaining and predicting process dynamics, guiding experimental design, and evaluating hypotheses [1, 2]. However, developing these models typically requires translating biological assumptions into mathematical formulations, a process that can introduce two primary sources of error. Firstly, the underlying biological hypotheses may be incomplete or inaccurate. Secondly, the conversion of these hypotheses into mathematical expressions may be faulty. In both instances, model selection techniques are valuable because they quantify the capacity of competing hypotheses—or their mathematical representations—to account for observed data [3].

Nowadays, there are several advanced workflows for developing mechanistic models of biological processes, which cover tasks such as model definition, simulation, parameterization, and uncertainty quantification. Model definition can be performed using standards like the Systems Biology Markup Language (SBML) [4], an Extensible Markup Language (XML)-based format capable of encoding biochemical reaction networks specifications which can be compiled to ordinary differential equations. Model simulation, parameterization, and uncertainty quantification can be specified using the Simulation Experiment Description Markup Language (SED-ML) [5] and the Parameter Estimation tables (PEtab) standard [6]. While SED-ML is an XML-based format, PEtab involves tab-separated value (TSV) files and YAML Ain't Markup Language (YAML) files. PEtab provides a flexible definition of measurement data, experimental conditions, observation models, and the parameters (including their priors) to be estimated. An ecosystem of software tools supports these formats, enabling seamless integration across diverse computing platforms and programming languages.

Some of these software tools also support model selection using information criteria (e.g., the Akaike Information Criterion (AIC) [7], corrected AIC (AICc) [8], or the Bayesian Information Criterion (BIC) [9]), Bayes factors [10], statistical tests including likelihood ratio tests [11], or cross-validation [12]. In fact, there are also software tools that allow for the selection of relevant features and the improvement of parameter identifiability through model reduction [13–17], facilitate model reduction for improved identifiability [18], or enable multi-model inference [19]. The most versatile tools appear to be Data2Dynamics (D2D) [20], dMod [21], pyPESTO [22], and BioModel Selection System 2 (BMSS2) [23]. However, none of these software tools provide comprehensive support for all methods, and the specification of model selection problems is often limited.

To address this critical gap, we introduce the PEtab Select `standard`, for the encoding of model selection problems in systems biology. The `standard` has been developed in a community effort and expands upon the widely adopted SBML [4] standard for describing mechanistic models and PEtab [6] for specifying parameter estimation problems. By providing a flexible encoding scheme for model selection, the `standard` allows for a compact description of thousands of candidate models using only

a few lines, thereby facilitating large-scale comparative analyses. Crucially, the `standard` retains compatibility with existing workflows by building on recognized community standards, eliminating the need for researchers to adopt entirely new toolchains.

Alongside the `standard`, we also provide a reference PEtab Select software `package`, which integrates seamlessly into established modeling workflows. The `package` can interface with popular PEtab-compatible systems biology model calibrators—such as COPASI [24] (via BasiCO [25]), Data2Dynamics [20], PEtab.jl [26], and pyPESTO [22]—leveraging the strengths of each. Implementing a truly general solution for model selection poses significant challenges, given the need for interoperable file formats and libraries compatible with diverse programming languages. The `package` addresses these challenges by building on community-endorsed standards and providing efficient built-in algorithms for model selection, including brute force, forward selection, backward selection, and FAMoS [27], the latter of which is designed for large model spaces. The subsequent sections detail the PEtab Select `standard` and `package`, and discuss how it can be readily combined with pre-existing calibration workflows to streamline model selection tasks. Fig. 1 illustrates this typical workflow.

Design and implementation

In this section, we outline the aims of PEtab Select and describe how they are fulfilled by the proposed new `standard` and accompanying software `package`. Subsequently, we detail the design and implementation of PEtab Select, which builds on the SBML and PEtab standards to ensure broad compatibility with existing workflows.

Scope

The model selection process begins by defining a set of admissible candidate models $\mathcal{M}$. In many cases, this set can be derived from a single, overarching "superset model" that encodes all processes under consideration; specific processes may then be removed or deactivated to yield distinct candidate models. When the range of competing hypotheses is broad, there may be multiple super models, each reflecting different conceptual assumptions or alternative sets of biological processes. For each of the candidate models in $\mathcal{M}$ there exists a corresponding parameter estimation problem that needs to be solved to compute the selection criteria. Importantly, to ensure the comparability of the values of the selection criteria, all estimation problems have to use the same dataset (and transformations of the dataset are not allowed). Model selection methods iterate over the set of admissible models until a stopping condition is met (e.g., no further improvement is achieved).

The PEtab Select ecosystem addresses these steps comprehensively. PEtab Select allows for (1) defining the model selection problem in the `standard` in terms of mathematical models and corresponding parameter estimation problems, (2) using a compatible calibration tool to fit the candidate models according to one of the model selection methods implemented in the `package`, and (3) receiving the final selection results in the `standard` format, and analyzing them with the `package` (Fig. 1).

The PEtab Select `standard`

The `standard` is an extension to the existing **PEtab** standard, utilizing the same TSV and YAML-based structure while introducing additional configuration options for model selection. Indeed, the `standard` preserves the core PEtab specification, which defines parameter estimation problem through tables of model parameters, experimental

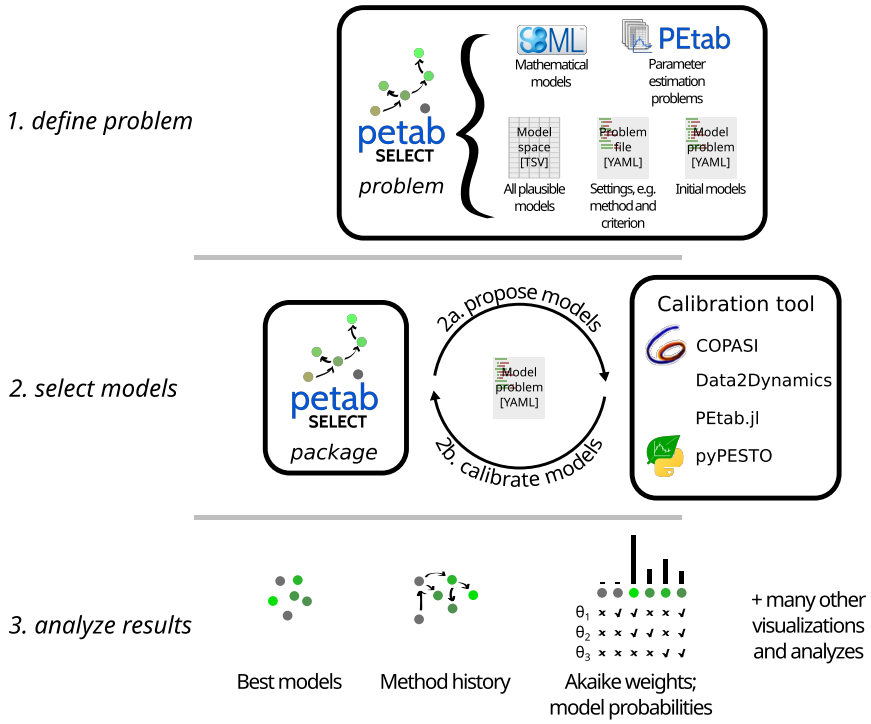


Fig 1. Model selection workflow with PEtab Select (1) The user defines the model selection problem, including the model space and the mathematical description of the set of possible process components, in the `standard`. (2) The user runs model selection via a *calibration tool* by sending the PEtab Select problem to the PEtab Select *package*, which initiates iterations of model selection. (2a) The package provides model candidates to the calibration tool based on the problem specification and the available calibration results. (2b) The calibration tool calibrates the proposed models and sends the calibration results to the `package`. Iterations of (2a) and (2b) continue until the `package` sends a termination signal to indicate the end of model selection. The results are provided in the PEtab Select model format and include all calibrated models with their calibration results. (3) The `package` can be used to perform subsequent analyses and visualizations of the results.

conditions, observables, and measurement data. Building on this, the `standard` adds a new layer dedicated to specifying the model space and relevant selection details (e.g., the subset of processes or reactions that can be toggled, the selection method to apply, and the selection criterion to use). This design ensures that existing PEtab workflows remain valid, while enabling that large and complex model spaces can be described in a concise manner (see Fig. 3 for an example). The resulting `standard` file formats are designed to be easy for both humans and machines to read and write, facilitating collaborative development and straightforward integration into broader systems biology pipelines.

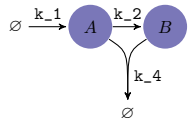
Model space (TSV) The model space is the set of models. In the `standard`, this can be defined as any combination of single models, and models derived from a PEtab problem with a superset mathematical model.

An example of defining a model space based on two superset models is given in Fig. 2. Each superset model has an associated mathematical model and parameter estimation problem specified in the `petab_yaml` column. This is customized to yield the

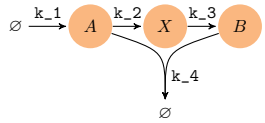
(a) The PEtab Select model space table.

Model subspace metadata		*Plausible parameter values*			
`model_subspace_id`	`model_subspace_petab_yaml`	`k_1`	`k_2`	`k_3`	`k_4`
`supersetA_models`	`problem1.yaml`	`0;estimate`	`0;estimate`	`0`	`0;estimate`
`supersetB_models`	`problem2.yaml`	`0;estimate`	`0;estimate`	`0;estimate`	`0;estimate`

(b) The superset model for the subspace `supersetA_models`.



(c) The superset model for the subspace `supersetB_models`.



(d) All models in the subspace `supersetA_models`.

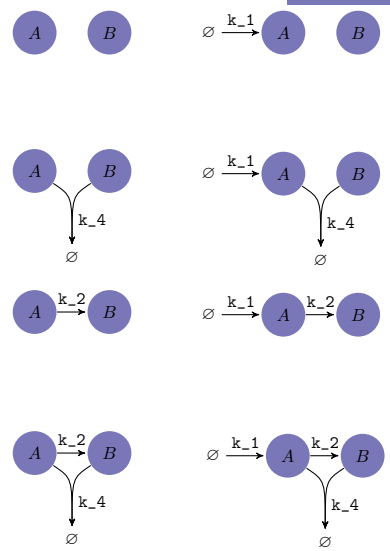


Fig 2. Example model space. (a) The model space table encodes model subspaces, which are rows of the model space table. Each model subspace has an ID and an associated superset PEtab problem, which includes the mathematical model. This superset problem is customized into single models according to the remaining columns, which contain semicolon-delimited lists of values that parameters can take in the subspace. (b,c) Each subspace is associated with a superset model, where all possible parameters are estimated. (d) Based on a superset model, candidate models are encoded by turning parameters on or off.

plausible submodels. The subspaces `supersetA_models` and `supersetB_models` encode 8 (Fig. 2d) and 16 models, respectively.

The model space file format is flexible to support a variety of use cases. A subspace with a single model can be specified by single values in the *Parameter values* columns, as opposed to semicolon-delimited lists. In this way, subspace `supersetA_models` could have alternatively been specified as 8 rows of single-model subspaces in the table. When dealing with multiple superset models that have different parameters (e.g., `k_3` is in `problem2.yaml` but not `problem1.yaml`), users may opt to use one model space table per superset model, to avoid sparsity or dummy parameters.

Note that for the example superset models in Fig. 2b and c, parameters are turned "off" by setting them to 0, which is not always the case. For example, a fold-change effect parameter might be turned off by setting it to 1.

Model files (YAML) Given mathematical models, parameter estimation problems, and the model space, the `package` can run model selection methods. The methods require calibrated models; hence, the `package` relies on compatible calibration tools to calibrate the models identified by the methods. The `package` sends uncalibrated models to calibration tools, the tools calibrate the models, and then the tools send the calibrated models back to the `package` (step 2 in Fig. 1).

The information is sent in the `standard` model file format, which is used to fully specify information at different steps of the workflow (Fig. 1): in step 1, the predecessor (initial) models of a method; in step 2, the models communicated between the `package` and calibration tools; and, in step 3, the results, such as all calibrated models or the best model. The information in the `standard` model file format includes: the properties (e.g. model subspace ID and estimated parameter IDs) that uniquely identify this model; the PEtab problem; and calibration results, such as the AIC value.

In a typical workflow, the model files (or equivalent Python objects) during steps 2 and 3 are automatically produced by the `package` and the PEtab-compatible calibration tool. The user can optionally specify predecessor models in step 1 manually, or use the PEtab Select software package to generate predecessor model(s).

Model selection problem file (YAML) As in PEtab, the `standard` uses a single YAML-based model selection "problem" file to combine all files of the model selection problem, such that compatible tools can load the model selection problem from a single file. The `standard` problem file also contains settings, including the model selection method to use, and the criterion by which models will be compared.

The PEtab Select `package`

The primary purpose of the `package` is to extend state-of-the-art calibration workflows, by providing an API that calibration tools can use to enable model selection capabilities. The `package` achieves this by performing all model selection functions, only requiring the calibration tool to calibrate the models that are generated by the `package` according to the chosen model selection method. Hence, users can select their preferred compatible tool that contains the calibration features they need.

Intended usage For this primary purpose, the user might never interact with the `package` directly, rather indirectly via their preferred calibration tool. Hence, this primary workflow (Fig. 1) is that users supply their specification files (model, PEtab, and PEtab Select files) to their calibration tool, then initiate model selection. Iterations of the chosen model selection algorithm are then coordinated manually by the user and/or automatically by the calibration tool, until the model selection problem is solved. Results are made available as calibrated models in the `standard` model format, that are stored in memory or on disk.

Calibration tools For a tool to perform model selection with PEtab Select, the tool needs to be able to calibrate PEtab problems and implement the `package` API. The `package` is written in Python 3 and the API is available via Python or the command line, such that calibration tools written in different languages can easily integrate PEtab Select. A test suite with several model selection problems is available to ensure the compatibility of calibration tools with the `package`.

We have worked with the developers of different state-of-the-art pipelines to integrate PEtab Select, such that users who are already familiar with one of the pipelines can easily add model selection to their workflow. Currently, PEtab Select is supported by COPASI [24] via BasiCO (Python) [25], Data2Dynamics (MATLAB) [1, 20], PEtab.jl (Julia) [26], and pyPESTO (Python) [22].

Other features The `package` provides several other features to enable flexible model selection. This includes automated summaries of the model selection run, custom or automated selection of predecessor (initial) models, limits on the number of models calibrated in any iteration, restarting of model selection runs with previous model

selection results, and object-oriented programming such that other methods and selection criteria can be easily implemented.

Model selection methods

Model selection methods can be broadly classified as global or local. In each algorithm, models are calibrated and compared by a model selection criterion. In this section, bold terms are methods and features that are available in the `package`.

Global methods To guarantee that the best model in the model space is identified, a global method is required. An example is the **brute-force** method, where every model is calibrated. However, if a model space is generated from a superset model with n components, then there are potentially 2^n plausible models in the model space. The brute-force method would therefore involve 2^n model calibrations, which may be computationally-infeasible.

Local methods To mitigate this, local methods such as stepwise regression can be employed. Local methods do not guarantee that the best model is found, but instead explore the model space efficiently to find good models. An example stepwise regression method is the **forward** method, which proceeds in iterations and begins with a predecessor model. In each iteration, all models with one additional estimated parameter, compared to the predecessor model, are calibrated. The best of these models is then selected as the predecessor model for the next iteration. Iterations proceed until the models in the current iteration do not improve on the current predecessor model, and then this predecessor model is selected as the best model. The **backward** method is similar but proceeds in the reverse direction, calibrating models with one fewer estimated parameters than the predecessor model.

The **FAMoS** method is an advanced local stepwise regression method that uses metaheuristics to change the local method employed over iterations [27]. Each FAMoS iteration involves the full run of all iterations of a simple method, such as the forward or backward method, or the lateral method, wherein simultaneous forward and backward changes are applied that keep the number of estimated parameters the same as the predecessor model. Between FAMoS iterations, a method switching scheme is employed to switch between local methods, based on the recent history of employed methods. For example, if the previous method was the backward method, and the current method is the forward method, then the next method is the lateral method. In PEtab Select, this method switching scheme is fully customizable. The FAMoS method also supports a "jump to most distant" local method. Full details of the FAMoS method are provided in Gabel et al. (2019) [27].

Model selection criteria All methods involve model comparisons according to a model selection criterion, in order to select the best or good models. Typically, these criteria try to simultaneously maximize both model goodness-of-fit and model parsimony. This involves a trade-off, since less parsimonious models tend to provide a better goodness-of-fit due to overfitting. Examples of such criteria include the **Akaike information criterion** (AIC) [7], **corrected AIC** (AICc) [8], **Bayesian information criterion** (BIC) [9]. The choice of criteria is nontrivial and left to the user. The AIC tends to select larger models for larger datasets, which may be undesirable. The AICc adds a small sample size correction factor to the AIC to mitigate overfitting, but this correction factor is derived based on univariate linear models with Gaussian-distributed residuals. The BIC is designed to select a model with a similar dimension as the

data-generating process but, like the AIC, the BIC is an asymptotic result and uses a Taylor-series approximation of the log-likelihood.

FAIR data principles

An important benefit of using PEtab Select is re-usability. For example, with PEtab Select, model selection results can be reproduced in different tools from a single specification of the model selection problem in the `standard`. The model selection results are also in the `standard`, such that they can be analyzed in a consistent manner, independent of the framework that produced them. For example, the `package` provides several plotting methods for results generated with any of the supported calibration tools.

The FAIR data principles [28] provide guidance on achieving human-out-of-the-loop re-usability, through the implementation of many principles that enable machine actionability. While the principles of findable and accessible data are out-of-scope here, PEtab Select implements several of the principles for interoperability and re-usability: the `standard` involves machine-readable file formats including YAML and TSV; the controlled vocabulary of keywords in the `standard` is documented and schemas for the `standard` YAML-based file formats are provided in the widely-adopted JSON Schema format; the `standard` provides sufficient information to fully reproduce a model selection problem; and, we utilize community standards where possible, by supporting the use of SBML and PEtab.

Results

PEtab Select enables model selection in existing workflows, each with unique tool-specific features

To assess the interoperability of **PEtab Select** with different software suites, we tested its functionality in four PEtab-compatible tools: **BasiCO** (built on **COPASI**), **Data2Dynamics**, **PEtab.jl**, and **pyPESTO**. Each tool successfully passed the tests in the test suite, confirming that they can parse and execute model selection tasks defined in the PEtab Select `standard`. While the software tools have a broad spectrum of overlapping features, there are a few distinctive characteristics.

BasiCO (COPASI) provides the possibility of custom evaluation functions in place of the standard information criteria. Moreover, its programmable interface enables model-specific optimizer settings, so callbacks or post-processing steps can be incorporated into the model selection workflow. This flexibility allows users to adjust parameters on a per-model basis or to visualize results automatically after each calibration step.

Data2Dynamics supports transferring parameter estimates from one calibrated model to another, potentially reducing computation time when new models are introduced. Additionally, users can define custom calibration routines to adjust the workflow before, during, or after parameter fitting. For example, the number of multi-starts in an optimizer can be dynamically set based on the size of the parameter space, or diagnostic plots can be automatically generated to assess calibration quality.

PEtab.jl integrates with the Julia SciML ecosystem, tapping into highly efficient ODE solvers provided by `DifferentialEquations.jl` [29] and automatic differentiation routines [30]. These capabilities enable rapid prototyping of novel approaches.

pyPESTO supports ensemble-based approaches, for instance using Akaike-weighted ensembles to combine the results from multiple candidate models. Its ready-to-use

post-processing tools also facilitate structured output of calibration results and automated generation of diagnostic figures, streamlining the analysis of potentially large model spaces.

PEtab Select enables reproducible model selection across workflows

To assess the comparability of results across workflows, we consider an exemplary model selection problem from the work of Blasi et al. (2016) [13]. It features a superset model where dynamics are governed by a reaction network with 16 state variables and 32 motif-specific reactions (Fig. 3a). Each reaction describes the conversion of one biochemical species into another. The rate constant of each motif-specific reaction is one of either a shared basal parameter or a motif-specific parameter. The model selection problem is to determine which reactions are sufficiently explained by the shared basal parameter, and which reactions require a motif-specific parameter instead. This model selection problem differs from the original publication in that we omit the additional 11 "site-specific" models in the original publication, which add a negligible computational cost and are not amongst the best models. Our model selection problem of 32 plausible motif-specific parameters therefore involves a model space of $2^{32} \approx 4.3$ billion plausible models.

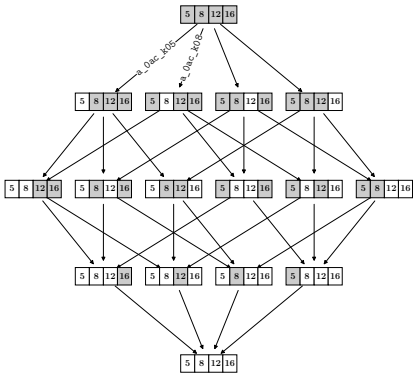
To concisely specify the model space with PEtab Select, the reaction rate constant for each reaction is formulated as the product of the shared basal parameter and the motif-specific parameter. The model selection problem then becomes whether to estimate motif-specific parameters, or to remove them by fixing them to 1. The corresponding PEtab Select model space table is shown in Fig. 3c. Details of the parameter estimation problem are specified by a single PEtab problem for the superset model (located at `petab_problem.yaml`), which is automatically customized by PEtab Select to represent the models that are visited during the search.

In the original publication, all ≈ 4.3 billion models were calibrated then compared, and they found that a model with 7 motif-specific parameters is the best (Fig. 3b and [13], Supplemental Information, Table S1).

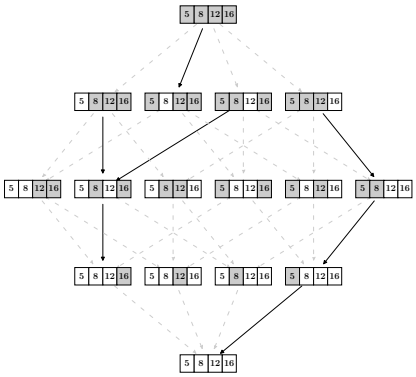
We used the FAMoS algorithm [27] initialized with 100 unique initial predecessor models that were randomly sampled from the model space; i.e., we repeated the algorithm 100 times, starting at different points in the model space (different initial sets of parameters). Usually, these initial predecessor models had about 16 parameters (Fig. 4b) due to the binomial nature of setting 32 parameters to be estimated or fixed to 1 with equal probability, and this is higher than the 7 parameters in the expected best model (Fig. 3b).

The initial experiment was to determine whether PEtab Select can solve this large model selection problem, and was conducted with pyPESTO as the calibration tool only. The model with the best criterion value across the 100 searches matched the best model from the original publication, and the FAMoS algorithm found it three times in the 100 repeats (Fig. 4a). In total, there were 83,125 ($\approx 0.00194\%$ of 2^{32}) model calibrations across all model starts. As the model starts were run independently without sharing results, some models were calibrated multiple times. However, PEtab Select supports user-provided calibration results, which means these duplicate calibrations are avoidable. The number of unique model calibrations was 58,858, which is $\approx 0.00137\%$ of the model space. In total, the 83,125 model calibrations took 655.8 CPU hours. Extrapolating to the full model space, this saved $\frac{2^{32}}{83125} \cdot \frac{655.8}{24 \cdot 365} = 3868$ CPU years; hence, this rendered the standardized approach computationally feasible, meaning that tailoring the simulation and optimization routines to the problem, as was done in the original publication, is not necessary.

(a) Superset model.



(b) Best model in Blasi et al. (2016).



(c) PEtab Select model space table for the Blasi et al. (2016) model selection problem.

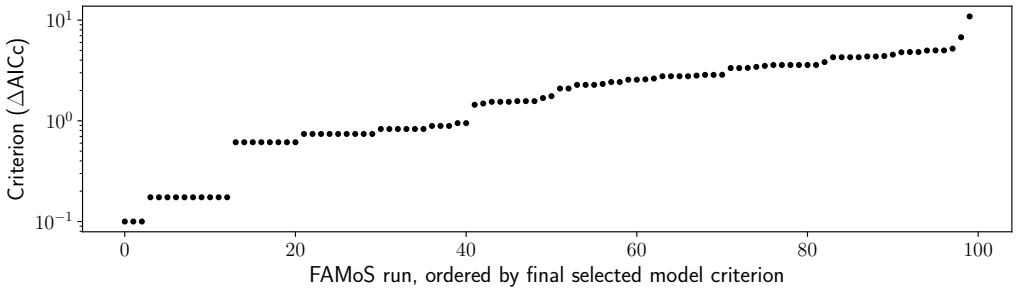
Model subspace metadata		*Plausible parameter values*		
`model_subspace_id`	`model_subspace_petab_yaml`	`a_0ac_k05`	`a_0ac_k08`	`...`
`M`	`petab_problem.yaml`	`1;estimate`	`1;estimate`	`...`

Fig 3. Blasi et al. (2016) [13] model space. (a) The superset model, with all possible motif-specific reactions. Nodes are state variables, which are combinations of the histone acetylation status at four different positions (5, 8, 12, and 16). Each position can be independently acetylated (white) or unacetylated (gray). The model selection problem is to identify motif-specific acetylation reaction rate constants (arrows), compared to a basal acetylation rate constant. (b) The best model identified in the study by Blasi et al. (2016). Dashed and solid arrows are reactions with basal and motif-specific rate constants, respectively. (c) The 32 hypotheses are whether the motif-specific (arrows in (a)) rate constants help explain the data (estimated) or not (fixed to 1). Only two parameterized hypotheses (`a_0ac_k05` and `a_0ac_k08`) are shown here for brevity, with the remaining 30 parameters represented by the ellipsis. This two-row table encodes the full model space of ≈ 4.3 billion models.

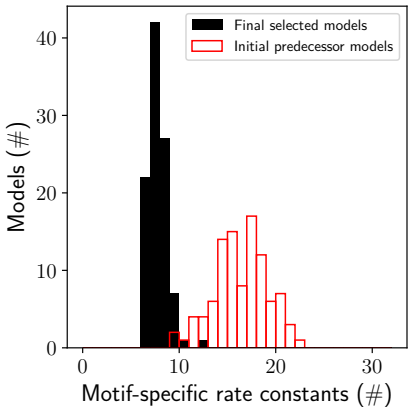
To assess reproducibility of model selection results generated with PEtab Select across all calibration tools, we re-ran one of the three searches that found the expected best model (Fig. 4c), with each calibration tool. The chosen search involved 339 model calibrations with pyPESTO, and this was also the case when the search was repeated with COPASI (BasiCO), Data2Dynamics, and PEtab.jl. With each of the tools, the expected best model was found, all tools calibrated a similar set of 339 models. Optimizer and other settings for each tool are provided in Appendix A.

Model selection for the considered problem is inherently challenging. The search space is vast and populated by many candidate models whose information-criterion values differ only within numerical precision, so minute floating-point variations render the ranking effectively non-deterministic across machines. Indeed, classical forward and backward selection mostly failed to arrive at the optimal solution. By contrast, the FAMoS algorithm [27] reproducibly converged on models with markedly better criterion scores. Although the reproduced single FAMoS search (with 339 model calibrations) takes slightly different trajectories through the model space on repeated executions within and between calibration tools (Fig. 5b-e), each trajectory delivers the same overall improvement within simulator and optimizer tolerances (Fig. 5a), with negligible deviation in criterion (ΔAICc < 0.01).

(a) The best models from 100 FAMoS runs.



(b) Model sizes across all 100 runs.



(c) One of three successful initial predecessor models.

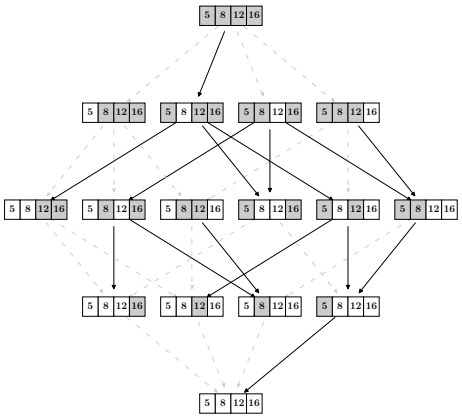


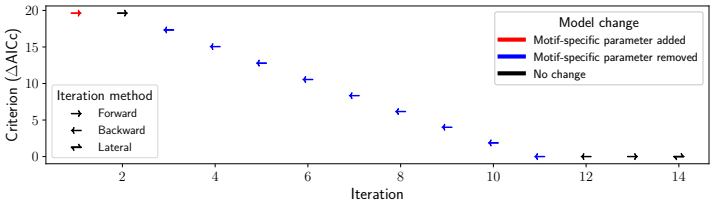
Fig 4. Multi-start FAMoS searches solve the Blasi et al. (2016) model selection problem. (c) "Successful" means that the search terminated at the best model seen across all searches, and matched the best model from the original publication (Fig. 3b). Dashed and solid arrows are reactions with basal and motif-specific rate constants, respectively.

Conclusion

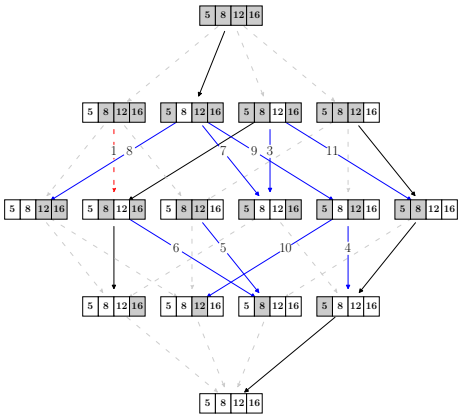
We present PEtab Select for (1) the concise specification, and (2) the efficient, reproducible solution, of model selection problems, which we achieve with a new standard and package. Currently, users can perform model selection with PEtab Select via COPASI (BasiCO), Data2Dynamics, PEtab.jl, and pyPESTO. We demonstrate the feasibility of this approach on a published model selection problem with $\approx 2^{32}$ models, by reproducing the result with each of these calibration tools.

There are several useful directions in which PEtab Select can be improved. Currently, PEtab Select only provides support for the SBML mathematical model format, as this is the only model format supported by PEtab. However, PEtab Select has no direct dependency on the model format. Hence, this is expected to improve with the next version of PEtab, which will support non-SBML model formats. Another improvement would be to implement calibration tool-specific features in the `package` directly, thereby making them available to all calibration tools. For example, the termination of a model selection iteration as soon as a better model is found. In this way, the PEtab Select design could provide opportunities to enable model selection features in multiple calibration frameworks, from single implementations. Additionally, thanks to the general Python API and CLI, users can readily implement model selection for unsupported

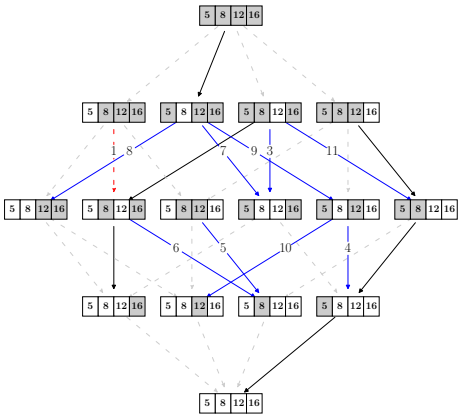
(a) The chosen successful FAMoS search, reproduced with all calibration tools.



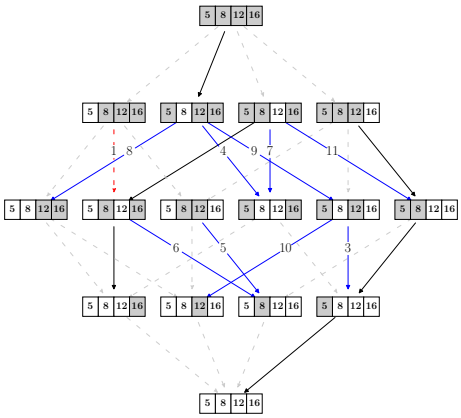
(b) COPASI (BasiCO) trajectory.



(c) Data2Dynamics trajectory.



(d) PEtab.jl trajectory.



(e) pyPESTO trajectory.

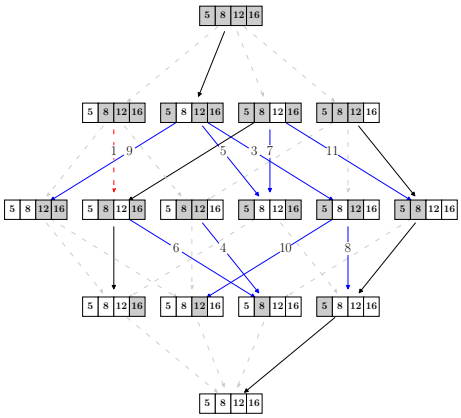


Fig 5. Reproducibility of model selection across calibration tools. (a) Model selection in each tool started with the same initial predecessor model (Fig. 4c) and ended at the same best model (Fig. 3b). The selected model at each iteration differed negligibly in criterion between tools (ΔAICc$<$0.01). (b, c, d, e) The trajectory through the model space taken with each calibration tool. Arrows are reactions. Dashed and solid arrows are reactions with basal and motif-specific rate constants, respectively, in the initial predecessor model. Motif-specific rate constants that were added through forward selection, or removed through backward selection, are red and blue, respectively, and these changes occurred on the iteration indicated by the number on the arrows.

calibration tools, model formats other than SBML, and problem formats other than PEtab. An example of this is provided in the documentation for the `package`, and a limitation is that this approach requires the generation of dummy PEtab files.

In addition to the implementation aspects, several opportunities for the improvement of model selection methodology were identified when working with the problem

published by Blasi et al. (2016) [13]. On one hand, the analysis revealed that reproducibility could be boosted by developing tailored strategies for handling the case that multiple models are equivalent in best criterion value (up to numerical noise). If performing a backward search, one possibility is to branch the backward search into multiple backward searches, one per equivalent model. On the other hand, a further improvement of search algorithms would be beneficial, in particular for large model spaces, e.g., an automatic tuning of metaheuristics in the FAMoS method across runs.

In summary, PEtab Select establishes, for the first time, a unified standard and reference implementation for large-scale, reproducible model selection. By decoupling the description of a model selection task from the calibration engine that solves it, the framework enables researchers to tackle search spaces of billions of candidates in whichever environment—COPASI, Data2Dynamics, pyPESTO, PEtab.jl, or future tools—best suits their workflows. This interoperability not only accelerates day-to-day practice but also lowers the barrier for methodological innovation, allowing new inference algorithms to be benchmarked immediately on realistic problems. We therefore expect PEtab Select to become a cornerstone for transparent, comparable, and scalable model selection across the life and physical sciences.

Acknowledgments

The authors are grateful to the many people who have provided feedback and contributions to the standard, package, and other aspects of PEtab Select, including participants of the HARMONY and COMBINE conferences (`https://co.mbine.org`).

This work was supported by the TRA Life and Health (University of Bonn) as part of the Excellence Strategy of the federal and state governments, the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy (project IDs 390685813 - EXC 2047 and 390873048 - EXC 2151) and through Metaflammation (project ID 432325352 – SFB 1454), the German Federal Ministry of Education and Research (BMBF) within the e:Med funding scheme (junior research alliance PeriNAA, grant no. 01ZX1916A), the Horizon Europe - ERC Consolidator Grant 2023 (grant agreement no. 101126146), the Federal Ministry of Education and Research (grant no. 031L0293C), and by the University of Bonn via the Schlegel professorship to J.H. FTB was supported by de.NBI, the German Network for Bioinformatics Infrastructure (W-de.NBI-016), and LIBIS. MC and SP were supported by the Swedish Research Council (VR2017-05117 and VR2023-04319) and the Swedish Foundation for Strategic Research (FFL15-0238). The work of NN, CK, and HB has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 499552394 – SFB 1597 Small Data.

PEtab Select developer group Severin Bang, Adrian Hauber, Svenja Kemmer, Daniel Lill, Lukas Refisch, Marcus Rosenblatt, Christian Tönsing, Franz-Georg Wieland.

References

1. Raue A, Schilling M, Bachmann J, Matteson A, Schelke M, Kaschek D, et al. Lessons Learned from Quantitative Dynamical Modeling in Systems Biology. PLOS ONE. 2013;8(9):e74335. doi:10.1371/journal.pone.0074335.

2. Villaverde AF, Pathirana D, Fröhlich F, Hasenauer J, Banga JR. A Protocol for Dynamic Model Calibration. Briefings in Bioinformatics. 2022;23(1):bbab387. doi:10.1093/bib/bbab387.

3. Burnham KP, Anderson DR, editors. Model Selection and Multimodel Inference. New York, NY: Springer; 2004.

4. Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, et al. The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models. Bioinformatics. 2003;19(4):524–531. doi:10.1093/bioinformatics/btg015.

5. Waltemath D, Adams R, Bergmann FT, Hucka M, Kolpakov F, Miller AK, et al. Reproducible Computational Biology Experiments with SED-ML - The Simulation Experiment Description Markup Language. BMC Systems Biology. 2011;5(1):198. doi:10.1186/1752-0509-5-198.

6. Schmiester L, Schälte Y, Bergmann FT, Camba T, Dudkin E, Egert J, et al. PEtab—Interoperable Specification of Parameter Estimation Problems in Systems Biology. PLOS Computational Biology. 2021;17(1):e1008646. doi:10.1371/journal.pcbi.1008646.

7. Akaike H. A New Look at the Statistical Model Identification. IEEE Transactions on Automatic Control. 1974;19(6):716–723. doi:10.1109/TAC.1974.1100705.

8. Sugiura N. Further Analysis of the Data by Akaike's Information Criterion and the Finite Corrections: Further Analysis of the Data by Akaike' s. Communications in Statistics - Theory and Methods. 1978;7(1):13–26. doi:10.1080/03610927808827599.

9. Schwarz G. Estimating the Dimension of a Model. The Annals of Statistics. 1978;6(2):461–464. doi:10.1214/aos/1176344136.

10. Kass RE, Raftery AE. Bayes Factors. Journal of the American Statistical Association. 1995;90(430):773–795. doi:10.1080/01621459.1995.10476572.

11. Tönsing C, Steiert B, Timmer J, Kreutz C. Likelihood-Ratio Test Statistic for the Finite-Sample Case in Nonlinear Ordinary Differential Equation Models. PLOS Computational Biology. 2023;19(9):e1011417. doi:10.1371/journal.pcbi.1011417.

12. Arlot S, Celisse A. A Survey of Cross-Validation Procedures for Model Selection. Statistics Surveys. 2010;4:40–79. doi:10.1214/09-SS054.

13. Blasi T, Feller C, Feigelman J, Hasenauer J, Imhof A, Theis FJ, et al. Combinatorial Histone Acetylation Patterns Are Generated by Motif-Specific Reactions. Cell Systems. 2016;2(1):49–58. doi:10.1016/j.cels.2016.01.002.

14. Milias-Argeitis A, Oliveira AP, Gerosa L, Falter L, Sauer U, Lygeros J. Elucidation of Genetic Interactions in the Yeast GATA-Factor Network Using Bayesian Model Selection. PLOS Computational Biology. 2016;12(3):e1004784. doi:10.1371/journal.pcbi.1004784.

15. Henriques D, Alonso-del-Real J, Querol A, Balsa-Canto E. Saccharomyces Cerevisiae and S. Kudriavzevii Synthetic Wine Fermentation Performance Dissected by Predictive Modeling. Frontiers in Microbiology. 2018;9:88. doi:10.3389/fmicb.2018.00088.

16. Mosbacher M, Lee SS, Yaakov G, Nadal-Ribelles M, de Nadal E, van Drogen F, et al. Positive Feedback Induces Switch between Distributive and Processive Phosphorylation of Hog1. Nature Communications. 2023;14(1):2477. doi:10.1038/s41467-023-37430-y.

17. Gupta S, Lee REC, Faeder JR. Parallel Tempering with Lasso for Model Reduction in Systems Biology. PLOS Computational Biology. 2020;16(3):e1007669. doi:10.1371/journal.pcbi.1007669.

18. Maiwald T, Hass H, Steiert B, Vanlier J, Engesser R, Raue A, et al. Driving the Model to Its Limit: Profile Likelihood Based Model Reduction. PLOS ONE. 2016;11(9):e0162366. doi:10.1371/journal.pone.0162366.

19. Theorell A, Nöh K. Reversible Jump MCMC for Multi-Model Inference in Metabolic Flux Analysis. Bioinformatics. 2020;36(1):232–240. doi:10.1093/bioinformatics/btz500.

20. Raue A, Steiert B, Schelker M, Kreutz C, Maiwald T, Hass H, et al. Data2Dynamics: A Modeling Environment Tailored to Parameter Estimation in Dynamical Systems. Bioinformatics. 2015;31(21):3558–3560. doi:10.1093/bioinformatics/btv405.

21. Kaschek D, Mader W, Fehling-Kaschek M, Rosenblatt M, Timmer J. Dynamic Modeling, Parameter Estimation, and Uncertainty Analysis in R. Journal of Statistical Software. 2019;88:1–32. doi:10.18637/jss.v088.i10.

22. Schälte Y, Fröhlich F, Jost PJ, Vanhoefer J, Pathirana D, Stapor P, et al. pyPESTO: A Modular and Scalable Tool for Parameter Estimation for Dynamic Models. Bioinformatics. 2023;39(11):btad711. doi:10.1093/bioinformatics/btad711.

23. Ngo RJK, Yeoh JW, Fan GHW, Loh WKS, Poh CL. BMSS2: A Unified Database-Driven Modeling Tool for Systematic Biomodel Selection. ACS Synthetic Biology. 2022;11(8):2901–2906. doi:10.1021/acssynbio.2c00123.

24. Hoops S, Sahle S, Gauges R, Lee C, Pahle J, Simus N, et al. COPASI—a COmplex PAthway SImulator. Bioinformatics. 2006;22(24):3067–3074. doi:10.1093/bioinformatics/btl485.

25. Bergmann FT. BASICO: A Simplified Python Interface to COPASI. Journal of Open Source Software. 2023;8(90):5553. doi:10.21105/joss.05553.

26. Persson S, Fröhlich F, Grein S, Loman T, Ognissanti D, Hasselgren V, et al. PEtab.Jl: Advancing the Efficiency and Utility of Dynamic Modelling. bioRxiv. 2025; p. 2025.04.30.651378. doi:10.1101/2025.04.30.651378.

27. Gabel M, Hohl T, Imle A, Fackler OT, Graw F. FAMoS: A Flexible and Dynamic Algorithm for Model Selection to Analyse Complex Systems Dynamics. PLOS Computational Biology. 2019;15(8):e1007230. doi:10.1371/journal.pcbi.1007230.

28. Wilkinson MD, Dumontier M, Aalbersberg IJ, Appleton G, Axton M, Baak A, et al. The FAIR Guiding Principles for Scientific Data Management and Stewardship. Scientific Data. 2016;3(1):160018. doi:10.1038/sdata.2016.18.

29. Rackauckas C, Nie Q. DifferentialEquations.Jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia. Journal of Open Research Software. 2017;5(1):15–15. doi:10.5334/jors.151.

30. Revels J, Lubin M, Papamarkou T. Forward-Mode Automatic Differentiation in Julia. arXiv. 2016;doi:10.48550/arXiv.1607.07892.

31. Hindmarsh AC, Brown PN, Grant KE, Lee SL, Serban R, Shumaker DE, et al. SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers. ACM Transactions on Mathematical Software. 2005;31(3):363–396. doi:10.1145/1089014.1089020.

32. Coleman TF, Li Y. On the Convergence of Interior-Reflective Newton Methods for Nonlinear Minimization Subject to Bounds. Mathematical Programming. 1994;67(1):189–224. doi:10.1007/BF01582221.

33. Coleman TF, Li Y. An Interior Trust Region Approach for Nonlinear Minimization Subject to Bounds. SIAM Journal on Optimization. 1996;6(2):418–445. doi:10.1137/0806023.

34. Fröhlich F, Sorger PK. Fides: Reliable Trust-Region Optimization for Parameter Estimation of Ordinary Differential Equation Models. PLOS Computational Biology. 2022;18(7):e1010322. doi:10.1371/journal.pcbi.1010322.

35. Fröhlich F, Weindl D, Schälte Y, Pathirana D, Paszkowski Ł, Lines GT, et al. AMICI: High-Performance Sensitivity Analysis for Large Ordinary Differential Equation Models. Bioinformatics. 2021;37(20):3676–3677. doi:10.1093/bioinformatics/btab227.

A Settings for the Blasi et al. (2016) example.

In this appendix, we describe the settings used with each tool to reproduce the Blasi et al. (2016) [13] model selection problem.

A.1 COPASI (BasiCO)

For model simulation we used the default LSODE solver with default tolerances (relative `1e-6` and absolute `1e-12`).

For model calibration we used some generations of Genetic Algorithm with stochastic ranking, followed by a local optimization using NL2SOL. Full details are provided in a notebook (`https://basico.readthedocs.io/en/latest/notebooks/FAMOS_ModelSelection.html`).

All computations were performed with BasiCO 0.83 and Python 3.10 on a Mac mini A2348 M1-Chip with 8 GB RAM and 256 GB SSD, or Python 3.13 on an Intel i9-14900K, 3200 MHz.

A.2 Data2Dynamics

For model simulation, we used the solver SUNDIALS CVODES 2.8.1 [31]. For gradient computations, we solved the sensitivity equations together with the original ODE system as described in [1]. The default simulation settings of the Data2Dynamics toolbox were used (relative and absolute tolerances equal to `1e-6`). The initial time for model equilibration was set to zero with the commands `arClearEvents; arSteadyState(1,1,1,{},0);`.

For model calibration, we used multi-start local optimization with 25 starts of an interior trust region reflective method for boundary constrained optimization problems, as described in [32, 33] and implemented in `lsqnonlin` of the MATLAB Optimization Toolbox. To ensure consistency of likelihood values with other calibration tools, we deactivated the Bessel correction by setting `ar.config.useFitErrorCorrection=0`.

For full details see the script online: `https://github.com/Data2Dynamics/d2d/blob/433ac71330da3c8778ddeda255605a2ae43577b6/arFramework3/Examples/ToyModels/PEtabSelect/TestCases/0009/runExample.m`

All computations were performed with Python 3.10.15, MATLAB Release R2022b and Data2Dynamics commit 433ac71 [1, 20] on a Dell Latitude 5420 with a single 11th Gen Intel(R) Core(TM) i5-1145G7 2.60GHz CPU, 24 GB RAM and 475 GB SSD storage.

A.3 PEtab.jl

For model simulation, we used the Rodas5P ODE solver from the `DifferentialEquations.jl` suite [29]. The default ODE solver settings were used (relative and absolute tolerances equal to `1e-8`). For gradient computations, we used forward-mode automatic differentiation from `ForwardDiff.jl` [30].

For model calibration, we used multistart local optimization with 50 starts of an interior trust region reflective method for boundary constrained optimization problems, as implemented in Fides v0.7.8 [34]. The Hessian update was the Broyden–Fletcher–Goldfarb–Shanno (BFGS) implementation in Fides.

All computations were performed with PEtab.jl v3.6.0 and Julia v1.11.3 on a single Intel(R) Xeon(R) Platinum 8168 2.70GHz CPU, with 8 GB rRAM and 25 GB SSD storage.

A.4 pyPESTO

For model simulation, we used the simulation tool AMICI v0.30.0 [35], that provides an interface to the backward differentiation formula implemented in the SUNDIALS CVODES solver [31], and we used its forward sensitivity analysis method for gradient computation. The default simulation settings were customized to have stricter relative tolerance (`1e-12`).

For model calibration, we used pyPESTO v0.5.4 [22]. We performed multi-start local optimization with 24 starts of an interior trust region reflective method for boundary constrained optimization problems, as described in [32, 33] and implemented in Fides v0.7.7 [34]. The Hessian update was the BFGS implementation in Fides.

All computations were performed on a single AMD EPYC 7F72 3.20 GHz CPU, with 1 TB RAM and 960 GB SSD storage.