yaml2sbml: Human-readable and -writable specification of ODE models and their conversion to SBML

Jakob Vanhoefer¹, Marta R. A. Matos², Dilan Pathirana¹, Yannik Schälte³ ⁴, and Jan Hasenauer¹ ³ ⁴

¹ Faculty of Mathematics and Natural Sciences, University of Bonn, Bonn, Germany
² The Novo Nordisk Foundation Center for Biosustainability, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark
³ Institute of Computational Biology, Helmholtz Zentrum München, Neuherberg, Germany
⁴ Department of Mathematics, Technical University Munich, Garching, Germany

DOI: 10.21105/joss.03215

Software
- Review
- Repository
- Archive

Editor: Kelly Rowland
Reviewers:
- @SirSharpest
- @marouenbg

Submitted: 09 April 2021
Published: 27 May 2021

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

Ordinary differential equations (ODE) models are used throughout natural sciences to describe dynamic processes. In systems biology, ODEs are mostly stored and exchanged using the Systems Biology Markup Language (SBML), a widely adopted community standard based on XML. The Parameter Estimation table (PEtab) format extends SBML to parameter estimation problems. A large number of software tools support simulation of SBML models and parameter estimation for PEtab problems. Specifying ODE models in SBML and parameter estimation problems in PEtab provides access to these tools. However, SBML is considered to be neither human-readable nor human-writable. An easy-to-use approach to construct the SBML/PEtab models tailored to ODE models will facilitate model generation.

In this contribution, we present yaml2sbml, a Python tool for converting ODE models specified in an easy-to-read and -write YAML file into SBML/PEtab. yaml2sbml comes with a format validator for the input YAML, a command-line interface (CLI) and a model editor to: 1) create an ODE model programmatically in Python that can then be saved as SBML, PEtab or YAML and 2) edit an ODE model previously encoded in YAML. Several examples illustrate the use of yaml2sbml on realistic problems.

Statement of need

ODE models have applications in many scientific fields, including biology (Hodgkin & Huxley, 1952; Lotka, 1920), epidemiology (Kermack et al., 1927), physics (Lorenz, 1963; Van der Pol & Van Der Mark, 1927), economics (Shone, 2002) or more recent neural networks (Chen et al., 2018).

SBML is a widely adopted community standard for specifying biological reaction networks (Hucka et al., 2003). sbml.org lists more than 100 software tools that accept SBML as their input format for dynamic model simulation, among them COPASI (Hoops et al., 2006), d2d (Raue et al., 2015), AMICI (Frölich et al., 2021) and dMod (Kaschek et al., 2019).

Model parameters can be estimated from data by formulating a likelihood function. Therefore, the system states must be mapped to measured quantities by observable functions, and a measurement noise model must be specified. The PEtab format was recently introduced to complement SBML by tab-separated value files specifying observables, measurements, experimental conditions, and estimated parameters (Schmiester et al., 2021). Currently 9 software
toolboxes support PEtab as an input format, among them COPASI, d2d, dMod and AMICI/pyPESTO. The PEtab documentation gives a complete and up-to-date list of tools.

Thanks to the aforementioned tools, model simulation or parameter estimation has become a matter of a few lines of code or clicks. However, ODE model definition is often a bottleneck, since constructing an SBML model from scratch is often tedious. Therefore, various approaches to facilitate model construction from text-based input formats or in code have been presented, as libsbml (Bornstein et al., 2008), SimpleSBML (Cannistra et al., 2015), MOCCASIN (Gómez et al., 2016), Antimony (Smith et al., 2009) and ScrumPy (Pooiman, 2006). MOCCASIN translates MATLAB code into SBML. Other tools have a text-based input format that is centered around chemical reactions and not around ODEs directly (e.g. ScrumPy), or only offer a text-based (Antimony) or only a Python-based way of defining SBML models (libsbml, SimpleSBML), but not both at the same time interchangeably. Neither of these tools offer PEtab support.

Here, we present a human-readable and -writeable format tailored to ODE models that is based on YAML and can be validated and translated to SBML and PEtab via the Python tool yaml2sbml and a CLI. Furthermore, yaml2sbml comes with a format validator and a Python-based model editor that allows to generate, import, extend and export a YAML model within code.

**Tool and Format**

Figure 1 gives an overview of the typical workflow for model generation and conversion using yaml2sbml.

Figure 1: Typical workflow for model generation and conversion using yaml2sbml. The ODE is written as YAML file using any text editor, the API, or the object-oriented model editor. Both can be used interchangeably. The conversion from YAML to SBML or PEtab can be performed in Python or by the CLI.

**YAML Format**

Building the input format on YAML allows to parse and validate the model easily, while keeping the simplicity of a text-based format (see Figure 1). The format is organized in the blocks for different model components.
- odes define states, right-hand sides and initial values (as numeric values or parameters).
- parameters define parameters and their values. Further optional keys, e.g. optimization bounds, are written to the PEtab parameter table.
- time specifies the name of the time variable.
- assignments dynamically assign a value to a variable. These are encoded as parameter assignment rules in the SBML file.
- functions define functions that can be called in other model parts.
- observables map ODE states to measurements. Observables can be encoded as parameter assignments in the SBML file or in the PEtab observable table.
- conditions define different experimental setups, e.g. specific inputs. Conditions are only encoded in the PEtab condition table and do not affect the SBML file.

For more details, we refer to the format specification.

Python Tool and Command-Line Interface

The Python tool yaml2sbml allows one to validate models specified in the YAML input format and convert them to SBML or PEtab via

```python
import yaml2sbml

# format validation
yaml2sbml.validate_yaml(yaml_file)

# SBML conversion
yaml2sbml.yaml2sbml(yaml_input_file, sbml_output_file)

# PEtab conversion
yaml2sbml.yaml2petab(yaml_input_file, PEtab_dir, model_name)
```

Validation is also performed internally before model conversion. libsbml (Bornstein et al., 2008) generates and validates the resulting SBML. The validator in the PEtab library checks the resulting TSV-files during conversion to PEtab.

Alongside its Python API, yaml2sbml comes with a CLI offering the same functionality via the commands yaml2sbml, yaml2petab, and yaml2sbml_validate.

yaml2sbmls model editor allows one to generate ODE models and programmatically add, delete, or modify model components. Further, the model editor allows one to import models from YAML and export them to YAML, SBML or PEtab.

Availability and Code Development

yaml2sbml is developed under the MIT license on GitHub. The tool is available on PyPI via pip install yaml2sbml. The documentation is hosted on readthedocs. Several Jupyter Notebooks contain examples covering all aspects of the tool. Code testing and continuous integration is performed via GitHub Actions.

Examples

Three notebooks use the Lotka-Volterra equations (Lotka, 1920) to introduce the different aspects of yaml2sbml: The Python toolbox, the CLI and the model editor. Another notebook showcases features of the input format as time-dependent or discontinuous right-hand sides.

The introductory examples are complemented by two more comprehensive examples of ODE models, which do not fit in the classical reaction network formulation for which SBML is intended.

The first application example considers the Chemical Master Equation (CME) (Gillespie, 1992), a stochastic model of (bio-)chemical processes. The Finite State Projection (FSP) truncates the infinite state space of the CME, yielding a finite-dimensional ODE (Munsky & Khammash, 2006). The example implements the FSP for a two-stage model of gene expression (Shahrezaei & Swain, 2008). yaml2sbml allows one to implement the 1000-dimensional ODE in less than 20 lines of code, by exploiting the rich problem structure.

The second application example considers a well-established ODE model of human glucose-insulin metabolism with 22 state variables (Sorensen, 1985). The Jupyter Notebook presents an implementation of the Sorensen model in the YAML format and uses the model editor to extend the preexisting YAML model to encode a patient-specific treatment.

Acknowledgement

The authors thank Elba Raimúndez for her help designing the logo.

Funding

J.V. was funded by the Federal Ministry of Economic Affairs and Energy (Grant no. 16KN074236) and the European Union’s Horizon 2020 research and innovation program (Grant No. 686282).

M.R.A.M. was funded by the Novo Nordisk Foundation (NNF10CC1016517 and NNF14OC0009473).

Y.S. was supported by the German Research Foundation (HA7376/1-1), and the German Federal Ministry of Education and Research (FitMultiCell; 031L0159A).

D.P. was funded by the Federal Ministry of Economic Affairs and Energy (Grant no. 16KN074236).

J.H. was funded by the Federal Ministry of Education and Research (Grant no. 01ZX1705) and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany’s Excellence Strategy - EXC 2151 - 390873048.

References


