ratesb_python: A Python Package for Analyzing Rate Laws in Biological Models

Longxuan Fan[1], **Joseph L. Hellerstein**[2], and **Herbert M. Sauro**[1]

1 Department of Bioengineering, University of Washington, 3720 15th Ave NE, Seattle, WA 98195, United States of America **2** eScience Institute, University of Washington, 3910 15th Ave NE, Seattle, WA 98195, United States of America

DOI: 10.21105/joss.07618

Software

- Review
- Repository
- Archive

Editor: Rocco Meli

Reviewers:

- @ayush9pandey
- @dilawar

Submitted: 22 September 2024
Published: 23 September 2025

Summary

ratesb_python is a Python package that analyzes mechanistic models of biological systems that consist of networks of chemical reactions like $2H_2 + O_2 \rightarrow 2H_2O$, with rate laws such as $k[h_2]^2[O_2]$ that describe the rate at which the reaction proceeds. The package focuses on rate laws of reactions, algebraic expressions that specify the rate at which reactants (e.g., H_2, O_2) are converted into products (e.g., O_2). ratesb_python analyzes rate laws to detect errors and warnings that affect the robustness and accuracy of models that use the SBML (Systems Biology Markup Language) community standard for model model descriptions (Hucka et al., 2003).

Statement of Need

Mechanistic models in systems biology are essential tools for simulating and understanding the intricacies of complex biological systems, and a wide variety of rate laws are used. One commonly used rate law is the *law of mass action* in which the reaction rate is proportional to the product of the concentrations of the reactants. For example, consider a reaction in which m molecules of A combine with n molecules of B to produce r molecules of C, or $mA + nB \rightarrow rC$. The mass action rate law is $k \times [A]^m \times [B]^n$, where $[x]$ is the concentration of x and k are constants.

The ratesb_python package evaluates rate laws against a library of predefined types to identify anomalies that may compromise the accuracy of mechanistic models. This process involves categorizing rate laws based on their mathematical characteristics and examining their performance within the context of the model. Such analysis enables ratesb_python to identify potential issues and display as errors and warnings, including discrepancies in reactant usage or abnormal reaction fluxes. For example, if the rate law provided for the reaction $2H_2 + O_2 \rightarrow 2H_2O$ is $k_1[H_2]^2[O_2][H_2O]^2$ (where k_1 is a constant), ratesb_python reports an error since there is no defined classification for the rate law since the products are included ($[H_2O]$).

Moreover, ratesb_python extends beyond symbolic comparisons employed by existing tools (such as SBMLKinetics (Xu, 2023)) by providing a **modular and extensible classification framework** that supports both predefined and user-defined rate laws. While symbolic methods rely on hardcoded heuristics and expensive algebraic simplification, ratesb_python introduces a structured approach that separates normalization (e.g., power-lowering, constant removal) from classification, enabling more consistent and accurate results. More importantly, the algorithm is not restricted to a set of pre-coded equations, thereby enabling more flexible customization: users can readily define their own rate laws and instruct ratesb_python to check whether a provided rate law matches a generalized or custom-defined functional form.

This functionality is especially beneficial for researchers who need to handle specialized or novel rate laws that do not necessarily fit the standard templates.

In simple scenearios-such as a Michaelis-Menten rate law $cell \times M \times V/(K+M)$ (where V and K are constants, M is the reactant, and $cell$ is the compartment) with one reactant and no products, both `ratesb_python` and symbolic methods produce similar outcomes. However, for more complex rate laws, such as a reversible Michaelis-Menten formulation. However, in more complex examples like a reversible michaelis menten rate law:

$$cell \times (V_{max}/K_{m1}) \times (S - P/K_{eq})/(1 + S/K_{m1} + P/K_{m2})$$

(constants: V_{max}, K_{m1}, K_{eq}, K_{m1}, K_{m2}; reactant: S; product: P; compartment: $cell$), symbolic methods often default to generic classifications, while `ratesb_python` maintains accuracy. This demonstrates that the package is particularly valuable for ensuring stability and correctness in real-world, biologically meaningful models.

In practice, this structured design makes `ratesb_python` less prone to misclassification in complex biological models. For example, in our benchmark comparison (see Experimental Comparison), `ratesb_python` correctly identified reversible Michaelis-Menten kinetics as RMM and RMMcat, whereas SBMLKinetics defaulted to the generic form FR. Similarly, for bidirectional mass-action variants, `ratesb_python` distinguished structurally distinct cases (BIDR21, BIDR11), while symbolic methods collapsed them into the same generic class (BIDR). These results highlight that, although runtime may vary across examples, `ratesb_python` consistently produces specific and accurate classifications, especially in cases where symbolic approaches lose detail. By emphasizing robustness and adaptability, the package enhances confidence in model correctness, which is often more critical than raw execution speed in systems biology workflows.

Software Description

`ratesb_python` analyzes rate laws to detect errors and warns about violations of best practices. Input to `ratesb_python` can be a file path to a model in the SBML or Antimony (Smith et al., 2009) formats, or a string in the Antimony format. The output is text and/or `Python` objects. Control over inputs and outputs is managed by `analyzer.py`.

Central to `ratesb_python` is the ability to classify rate laws according to widely used types such as: mass action, Michaelis-Menten, and zeroth order kinetics. `ratesb_python` relies heavily on approaches employed in SBMLKinetics (Xu, 2023), which uses the `sympy` package to do symbolic analysis of rate laws. However, `ratesb_python` refines and extends these approaches by using a randomized polynomial identity test, providing a more efficient and customizable framework for classifying rate laws. This functionality is complemented by `custom_classifier.py`, which offers users the flexibility to define and classify rate laws via a structured JSON format. This adaptability is crucial for tailoring the tool to specific research requirements, highlighting `ratesb_python`'s commitment to user-defined customization. Default classifications are detailed in `default_classifier.json`.

Error and warning messages generated during the analysis are systematically managed within `messages.json`, ensuring users are well-informed of any issues detected during the examination process. The results of these analyses are succinctly presented through the `Results` class in `results.py`, providing users with a clear description of the findings.

Error Code Rationale and Organization
`ratesb_python` employs *grouped error codes* to systematically guide users in identifying and addressing issues with rate law classification. This ensures that common issues (e.g., missing reactants, invalid product usage, or typographical mistakes) are clearly distinguished from more severe problems (e.g., completely undefined rate laws). The codes also serve developers by providing a standardized mechanism for adding new checks or extending existing ones. Detailed

explanations of these codes, along with examples, can be found both in the `messages.json` file and in the official documentation. By grouping related issues under specific ranges of codes, `ratesb_python` makes it easier for users to trace potential errors to their underlying causes and for developers to introduce new error or warning categories without breaking the existing structure.

Below is a summary of the error and warning codes along with their brief descriptions:

Code	Type	Brief Description
1-2	Errors	Issues with missing rate laws or expected reactants.
1001	Warning	Numeric-only rate law.
1002	Warning	Rate law unrecognized.
1003-1004	Warning	Flux relationship issues with reactants and products.
1005	Warning	Missing boundary species reactant.
1006	Warning	Non-constant parameters in rate law.
1010	Warning	Products in irreversible reaction rate law.
1020-1022	Warning	Naming conventions for parameters not followed.
1030-1037	Warning	Issues with ordering and formatting conventions in rate laws.
1040-1044	Warning	Annotations not following recommended SBO terms.

Error and warning messages to aid in rate law analysis.

Experimental Comparison

To evaluate classification stability and accuracy, we conducted a comprehensive benchmark comparing `ratesb_python` against `SBMLKinetics` on 13 SBML models encompassing both simple and complex biological systems. Our benchmark includes single-reaction models covering standard rate law types (mass action, Michaelis-Menten, reversible kinetics) as well as multi-reaction models representing realistic biological pathways with mixed kinetics. This approach tests not only individual rate law classification but also the tools' ability to maintain accuracy across complex models with multiple diverse reactions.

Model	Reactions	RatesB Acc	SBMLK Acc	RatesB Time(s)	SBMLK Time(s)
bidirectional_bidr21.xml	1	100.00%	100.00%	0.103	0.016
bidirectional_bidr_a11.xml	1	100.00%	50.00%	0.021	0.013
branched_network.xml	5	100.00%	100.00%	0.233	0.026
enzyme_cascade.xml	3	100.00%	0.00%	0.353	0.052
mass_action_undr1.xml	1	100.00%	100.00%	0.005	0.010
mass_action_undr2.xml	1	100.00%	50.00%	0.010	0.010
metabolic_cycle.xml	5	80.00%	30.00%	0.351	0.985
michaelis_menten_mm.xml	1	100.00%	100.00%	0.113	0.014
michaelis_menten_mmcat.xml	1	100.00%	100.00%	0.113	0.058
mixed_pathway_1.xml	3	100.00%	100.00%	0.171	0.021
reversible_mm_rmm.xml	1	100.00%	0.00%	0.175	1.004

Model	Reactions	RatesB Acc	SBMLK Acc	RatesB Time(s)	SBMLK Time(s)
reversible_mm_rmm-cat.xml	1	100.00%	0.00%	0.226	0.021
reversible_path-way.xml	4	75.00%	25.00%	0.139	0.028
OVERALL	**28**	**92.86%**	**55.36%**	**2.011**	**2.259**

The results demonstrate `ratesb_python`'s superior classification stability, achieving **92.86% accuracy** compared to SBMLKinetics' **55.36% accuracy** across 28 individual reactions. This 37.5 percentage point advantage becomes particularly pronounced in complex models: while both tools perform similarly on simple cases (e.g., basic mass action and Michaelis-Menten), `ratesb_python` maintains high accuracy on challenging cases like enzyme cascades (100% vs 0%) and reversible Michaelis-Menten kinetics (100% vs 0%), where SBMLKinetics defaults to generic "FR" (fraction) classifications that lack biological specificity.

Crucially, this stability advantage is most valuable in realistic biological modeling scenarios where models contain multiple reaction types. The benchmark includes complex multi-reaction models (branched_network.xml, metabolic_cycle.xml, reversible_pathway.xml) that represent the types of systems biologists actually work with. In these cases, `ratesb_python`'s structured classification approach consistently identifies specific rate law subtypes (BIDR21 vs BIDR11, RMM vs RMMcat), while SBMLKinetics' symbolic methods often lose precision and default to broader categories. This reliability in complex scenarios makes `ratesb_python` particularly valuable for researchers who require accurate rate law identification for model validation and biological interpretation.

Integration with Other Tools and API Capabilities

`ratesb_python` is designed as a flexible, modular API and standalone tool, enabling integration with various systems biology tools to facilitate rate law analysis in biological modeling projects. Its development in `Python` ensures compatibility with prevalent scientific computing tools, allowing it to be added to existing systems or tailored for specific applications. It works well with tools that are widely used in the SBML community, such as `libantimony` and `libsbml`. Additionally, `ratesb_python` serves an educational purpose, offering a practical tool for computational biology courses where students can learn about rate law analysis by interacting with and modifying the API.

Future Work

Future developments for `ratesb_python` include enriching the library of checks, optimizing the performance of classification algorithms improving visualization of classification outcomes, and expanding the benchmark set to cover a broader spectrum of biologically relevant models. In particular, plotting classification correctness and stability against model complexity will provide a clearer picture of where existing tools struggle. Performance optimization of classification algorithms remains a secondary goal, but the primary focus is on strengthening correctness, robustness, and usability for researchers.

Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., Arkin, A. P., Bornstein, B. J., Bray, D., Cornish-Bowden, A., Cuellar, A. A., Dronov, S., Gilles, E. D., Ginkel, M., Gor, V., Goryanin, I. I., Hedley, W. J., Hodgman, T. C., Hofmeyr, J.-H., … SBML Forum:, the rest of the. (2003). The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, *19*(4), 524–531. https://doi.org/10.1093/bioinformatics/btg015

Smith, L. P., Bergmann, F. T., Chandran, D., & Sauro, H. M. (2009). Antimony: A modular model definition language. *Bioinformatics*, *25*(18), 2452–2454. https://doi.org/10.1093/bioinformatics/btp401

Xu, J. (2023). SBMLKinetics: A tool for annotation-independent classification of reaction kinetics for SBML models. *BMC Bioinformatics*, *24*(1), 248. https://doi.org/10.1186/s12859-023-05380-3