BioSimulators: a web-based registry of simulation engines and services for multiscale modeling

Bilal Shaikh[1], Lucian P. Smith[2], Dan Vasilescu[3], Gnaneswara Marupilla[3], Mike Wilson[3], BioSimulators Consortium (names to be inserted here), Arthur P. Goldberg[1], James C. Schaff[3], Michael L. Blinov[3], Herbert M. Sauro[2], Ion I. Moraru[3] and Jonathan R. Karr[1,*]

[1]Icahn School of Medicine at Mount Sinai, New York, NY 10029, US, [2]University of Washington, Seattle, WA 98105, US, and [5]University of Connecticut School of Medicine, Farmington, CT 06030, US.

ABSTRACT

Computational models have great potential to accelerate bioscience, bioengineering, and medicine. One promising way to build models is to combine submodels of multiple subsystems and scales that involve multiple modeling methods. However, it remains challenging to combine simulations because these methods remain siloed by different software tools. For example, each tool must be executed through a distinct interface using particular model formats. To help investigators find and use these tools, we developed BioSimulators (https://biosimulators.org), a central registry of the capabilities of simulation tools and consistent Python, command-line, and containerized interfaces to them. The foundation of BioSimulators is standards such as CellML, SBML, SED-ML, and the COMBINE archive format, and tools for validating that simulation projects and tools use these standards consistently. To help modelers find tools for particular projects, we have also used the registry to develop recommendation services. Together, we anticipate that BioSimulators will help modelers exchange, reproduce, and combine simulations.

INTRODUCTION

Sophisticated computational models that can predict biological phenomena have great potential for bioscience, bioengineering, and medicine. For example, whole-models could help scientists understand the origin of behavior, help engineers design biofactories, and help clinicians personalize medicine (1, 2). Due to the complexity of biology, such models often need to integrate multiple subsystems across multiple scales, requiring collaborations among teams (3, 4).

Over the last 25 years, researchers have developed numerous methods and tools for simulating various subsystems and scales. For example, COBRApy (5) and COPASI (6) can execute constraint-based and kinetic simulations of metabolic and signaling networks, respectively.

Toward combining simulations, the community has developed several resources for sharing several types of models. For example, formats such as CellML (7) and SBML (8); libraries for these formats such as libSBML and libNeuroML; and repositories such as BioModels (9) and ModelDB (10) help investigators share several types of models.

More recently, investigators have initiated similar efforts to share several types of simulations. For example, the Simulation Experiment Description Language (SED-ML) (11), the COMBINE archive format (12); the Kinetic Simulation Algorithm Ontology (KiSAO) (13), and the JWS Online repository (14) can be used to share kinetic simulations.

Furthermore, several resources for sharing several types of simulation tools are also available. For example, the SBML Software Guide and test suite can help investigators find tools for particular SBML models. In addition, container registries, such as BioContainers, can be used to share simulators.

Despite this progress, sharing, reusing, and combining simulations remains difficult. One major reason is that it is difficult to find, obtain, and use appropriate tools for particular systems and scales. For example, the existing registries of tools each focus on a subset of simulation methods, and these registries have limited quality controls to ensure their accuracy. Furthermore, each tool must be obtained from a different location, installed via a different process, and executed via a different API using different model formats.

To accelerate the development of multiscale simulations, we developed BioSimulators, a central registry for the capabilities of simulation tools (e.g., supported model formats, modeling frameworks, and simulation algorithms) and consistent Python, command-line, and containerized interfaces to these tools. Currently, BioSimulators encompasses 55 tools for 13 model formats, 14 modeling frameworks, and 91 simulation algorithms (Tables S1-S3), and consistent interfaces to 21 of these tools (Tables S4,S5). To help investigators find appropriate tools, we have also used this registry to develop services for recommending specific algorithms and tools for particular systems and scales.

To simplify the discovery, installation, and use of simulation tools, BioSimulators is based on an integrated set of formats, ontologies, and quality controls (Figure 1). BioSimulators uses the Open Container Initiative format (OCI; commonly known as Docker) to encapsulate simulation tools and a new schema to capture their capabilities. The input to each tool is a COMBINE archive which contains SED-ML files that describe simulations of models in formats such as SBML with algorithms described using KiSAO. The outputs of each tool are HDF5 and PDF files that capture the results and visualizations of simulations. To ensure these resources are used consistently, we also developed tools for integrated validation of simulation projects and tools (Figure 2a). On top of BioSimulators, we have also developed runBioSimulations and BioSimulations, user-friendly web applications

*To whom correspondence should be addressed. Tel: +1 212-659-8973 (JRK); Email: karr@mssm.edu

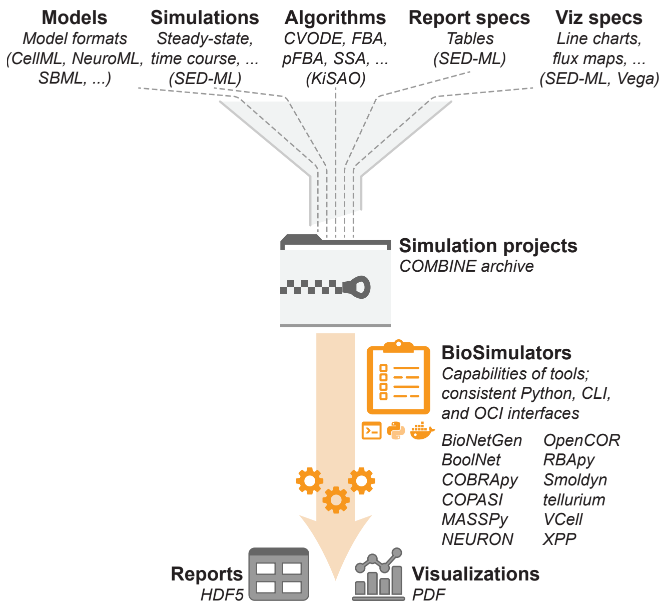
Models
Model formats (CellML, NeuroML, SBML, ...)
Simulations
Steady-state, time course, ... (SED-ML)
Algorithms
CVODE, FBA, pFBA, SSA, ... (KiSAO)
Report specs
Tables (SED-ML)
Viz specs
Line charts, flux maps, ... (SED-ML, Vega)
Simulation projects
COMBINE archive
BioSimulators
Capabilities of tools; consistent Python, CLI, and OCI interfaces
BioNetGen
BoolNet
COBRApy
COPASI
MASSPy
NEURON
OpenCOR
RBApy
Smoldyn
tellurium
VCell
XPP
Reports
HDF5
Visualizations
PDF

Figure 1. BioSimulators simplifies simulation by abstracting simulation projects as COMBINE archives and encapsulating simulation tools as containerized command-line interfaces and specifications of their capabilities. This makes it easier to execute a broad range of simulations.

for using BioSimulators to execute and share simulations and visualizations of their results (15) (Figure 2c).

Below, we summarize the key features of BioSimulators, outline the architecture of BioSimulators, delineate several use cases for BioSimulators, and describe how we aim to continue to accelerate multiscale simulation.

KEY FEATURES

The central features of BioSimulators are streamlined abilities to find, obtain, and use simulation tools for a broad range of modeling frameworks, formats, and algorithms.

Streamlined discovery of appropriate tools for projects

To help investigators find appropriate tools, BioSimulators provides a central database of the capabilities of simulation tools. This includes the model formats, modeling frameworks, simulation algorithms, and observables that each tool supports; the parameters of each algorithm; and the data type of each parameter, as well as meta data, such as the license of each tool. Where possible, this information is captured using ontologies such as EDAM, KiSAO, SBO, and SIO. This ensures that simulation tools are described consistently.

To further help investigators find tools, we have also used the capabilities of each tool and relationships among simulation algorithms encoded into the KiSAO ontology to develop several recommendation services. For example, we have developed a web form that can recommend simulation tools that can execute particular projects.

Streamlined acquisition and installation of simulators

To make it easier to obtain and install simulation tools, BioSimulators saves an OCI image for each version of each containerized tool. This ensures that investigators can use a single program, such as Docker Desktop, to easily obtain and install any version of any tool. To ensure that investigators can use these images in high-performance computing (HPC) environments, BioSimulators tests that these images are compatible with the Singularity format, which can be run in HPC environments. Similarly, the Python APIs and command-line programs for simulation tools can be installed consistently from the PyPI repository.

Streamlined execution of simulation tools

To simplify the execution of simulations, each containerized tool provides the same command-line interface. These interfaces capture the simulation project that should be executed and the location where its outputs (reports and visualizations) should be saved. This enables investigators to use multiple simulation tools simply by switching the their image ids. We anticipate this will help investigators work with a broader range of simulations, especially trainees, experimentalists, and peer reviewers. Most of the simulation tools with containerized interfaces also provide Python APIs. These APIs provide consistent, flexible low-level simulation capabilities. Currently, we are helping several groups use these APIs to develop interactive tools for research and education.

Accurate and up-to-date information about simulators

To ensure that the interfaces to simulation tools are consistent and that their capabilities are accurate, BioSimulators extensively reviews each version of each tool. The first version of each tool that is submitted to BioSimulators is reviewed by our team. Subsequent versions of tools are validated by an automated test suite that we developed. This test suite uses the simulation tool to execute a set of example COMBINE archives and checks that the tool produced the expected results. To enable software developers to keep BioSimulators up to date, BioSimulators provides an API that developers can use to automatically submit each version of their tools. Together, we anticipate that this API and validation will enable us to keep BioSimulators up-to-date and accurate.

METHODS

Consistent representation of simulation projects and tools

The foundation of BioSimulators is a set of formats, ontologies, and specifications for consistently representing simulation projects (one or more simulations of one or more models using one or more algorithms), the outputs of simulation projects (data sets and visualizations of simulation results), simulation tools, and the capabilities of simulation tools (supported model formats, modeling frameworks, and simulation algorithms). Where possible, these conventions embrace existing community resources. Where needed, we have filled gaps within and between these resources with additional formats, ontology terms, and specifications.

BioSimulators uses the COMBINE archive format to encapsulate all of the files that constitute a simulation project. Within COMBINE archives, BioSimulators uses community formats such as BNGL, CellML, GINML, NeuroML/LEMS, RBA XML, SBML, Smoldyn, and the XPP ODE to describe models and SED-ML to describe analyses of these models, such as simulations of time courses and steady-states. Within SED-ML, BioSimulators uses the KiSAO ontology to describe the algorithms and algorithm parameters for these analyses. To enable investigators to describe a broad range of simulations, we significantly expanded the KiSAO ontology and filled several gaps in SED-ML (11).

To consistently capture the outputs of simulation projects, we developed schemas for encoding the results of simulations into HDF5 files and encoding logs of the execution of simulation projects into YAML. BioSimulators uses the PDF format to capture visualizations of simulation results.

To enable investigators to execute simulation tools consistently, we developed conventions for Python APIs, command-line programs, and containerized command-line programs for simulation tools. To help investigators find specific tools for particular projects, we developed a schema for capturing the capabilities of simulation tools. This schema uses the EDAM, SBO, KiSAO, SIO, and other ontologies to capture the model formats, modeling frameworks, simulation algorithms, and simulation observables that each tool supports. We similarly helped expand these ontologies to enable investigators to capture a broad range of tools.

More information about these conventions is available in Section S2 and at https://docs.biosimulations.org.

Standardized interfaces to simulation tools

We developed most of the Python, command-line, and containerized interfaces to simulation tools by combining simulation tools, such as COBRApy and COPASI, with BioSimulators-utils, a library that we developed for orchestrating the execution of COMBINE archives. Briefly, BioSimulators-utils executes each simulation task in each SED-ML file in a COMBINE archive by (1) resolving the model for the task, (2) modifying the model according to the changes specified in SED-ML, (3) using KiSAO to determine the most similar simulation algorithm that the simulation tool implements to the algorithm specified for the task, (4) translating this algorithm and its specified parameters into the corresponding method of the simulation tool and its arguments, (5) executing this method with these arguments, (6) collecting the results of this method, (7) and using these results to generate the reports and plots specified in the SED-ML files. This modular design minimizes the effort needed to create standardized interfaces to simulation tools. The use of KiSAO to automatically identify suitable alternative algorithms enables investigators to both (a) use SED-ML to precisely record the algorithms they used to execute simulations with one tool and (b) execute the SED-ML files with additional tools that implement similar algorithms. Section S4 contains more information about BioSimulators-utils.

Recommendations of algorithms and simulation tools

To help investigators navigate the sea of simulation formats, methods, and tools, we developed several interfaces for recommending resources, including (a) an interactive table for searching and filtering the registry for tools that support specific combinations of resources; (b) a web form for obtaining a list of tools which implement algorithms similar to a desired algorithm, sorted by the maximal similarity of each simulators' algorithms to the desired algorithm; and (c) a web form for identifying the simulation tools which can execute a specific simulation project using algorithms which are at least as similar to the specified algorithms as a selected degree of similarity. Briefly, we implemented these recommendation services by (a) using parent-child and other relationships to encode similarities among algorithms into the KiSAO ontology, (b) using these relationships to query the ontology for groups of similar algorithms, (c) manually assigning each group a degree of similarity, (d) using BioSimulators-utils to determine the model formats and simulation algorithms specified for each project, (e) using the registry of simulation tools to determine the capabilities of each simulation tool, and (f) combining this information to determine the maximal degree of similarity at which each simulation tool can execute a set of model formats and simulation algorithms. More information is available in Section S4.

Validation of simulation projects and tools

To ensure that BioSimulators' conventions are used consistently and to quickly alert users to issues, we developed a tool for integrated validation of COMBINE archives (model, SED-ML, and meta data files), as well as tools for validating the new schemas described above for simulation results, logs of the execution of simulations, and the capabilities of simulation tools. This included developing the first validation rules for SED-ML. For example, our tool for validating simulation projects checks that each SED-ML file is consistent with the SED-ML schema and that each observable of each simulation references a valid variable of the corresponding model. To make these validation tools easy to use, we developed several interfaces to the tools, including web forms, a REST API, a command-line program, and a Python API.

Similarly, we also developed a test suite for checking whether simulation tools execute projects consistently with BioSimulators' conventions. Briefly, the test suite executes simulation tools with a set of test COMBINE archives and checks that they produced the expected outputs. These test archives enable the test suite to probe support for all of BioSimulator's conventions, including all of the features of the COMBINE archive format and SED-ML. To enable us to test tools involving a broad range of model formats and simulation algorithms, the test suite computationally generates the majority of the test archives from a modest number of manually curated archives. This design makes it easy to expand the test suite to additional model formats and methods. To make the test suite easy to use, we developed a command-line interface. We have also deployed the test suite using GitHub issues and actions. This enables the test suite to be run simply by submitting an issue to the BioSimulators GitHub repository, in which case the results of the test suite are communicated as messages to the issue. GitHub issues also enable our team to monitor problems that developers are encountering and help them. More information about these validation tools is available in Section S3.

Submission of simulation tools to the registry

Developers can submit simulation tools to the registry by creating an issue in the BioSimulators GitHub repository with a URL for the specifications of their tool. Each version of each tool that provides a containerized interface is then automatically run with our test suite and only committed to the registry if it passes each test. In addition, we manually review the first version of each tool submitted to BioSimulators. This manual review enables us to check aspects of tools that are challenging to test programmatically, such as the completeness of their specifications. This combination of machine and human review enables us to review each version of each tool with minimal effort.

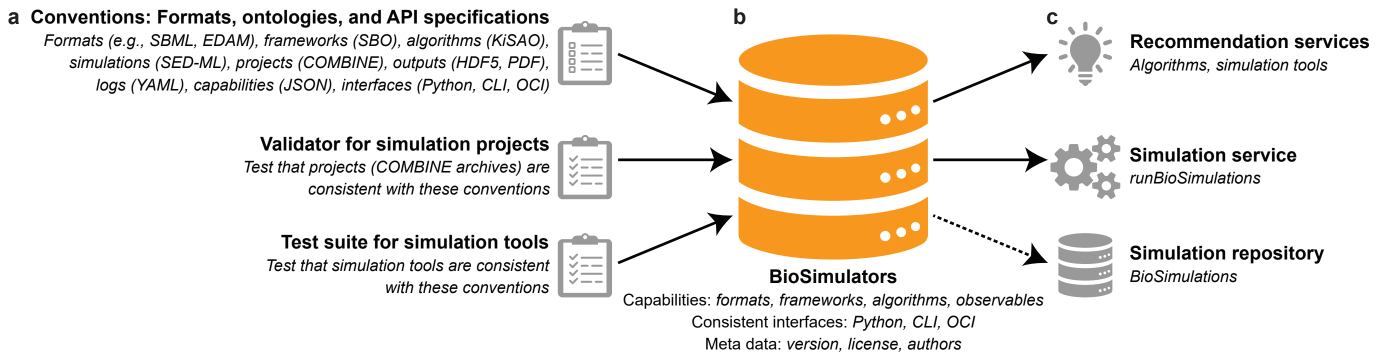
a Conventions: Formats, ontologies, and API specifications
Formats (e.g., SBML, EDAM), frameworks (SBO), algorithms (KiSAO), simulations (SED-ML), projects (COMBINE), outputs (HDF5, PDF), logs (YAML), capabilities (JSON), interfaces (Python, CLI, OCI)
Validator for simulation projects
Test that projects (COMBINE archives) are consistent with these conventions
Test suite for simulation tools
Test that simulation tools are consistent with these conventions
b
BioSimulators
Capabilities: formats, frameworks, algorithms, observables
Consistent interfaces: Python, CLI, OCI
Meta data: version, license, authors
c
Recommendation services
Algorithms, simulation tools
Simulation service
runBioSimulations
Simulation repository
BioSimulations

Figure 2. Overview of the BioSimulators ecosystem. The foundation of BioSimulators (**b**) is an integrated set of formats, ontologies, and specifications for simulation projects and simulation tools, and tools for checking that these conventions are used consistently (**a**). These conventions make it easier to work with multiple types of simulations. To further help investigators find and run simulation tools, we have also developed user-friendly services for recommending tools, executing simulations, and visualizing the results of simulations. In addition, we are developing a repository for sharing projects, their results, and visualizations of these results (**c**).

We chose to use GitHub issues to manage the submission of simulation tools for two reasons. First, this enables the community to see how each tool was validated. Second, this provides developers an API for programmatically submitting tools. Importantly, this API makes it easy for developers to keep BioSimulators up-to-date with the latest versions of their tools. For example, developers can use this API and GitHub actions to automatically submit each release of their tools to BioSimulators. Currently, half of the containerized tools registered with BioSimulators automatically release each version to BioSimulators.

DESIGN, IMPLEMENTATION, AND DEPLOYMENT

BioSimulators is composed of a set of conventions for consistently representing simulation projects and simulation tools; a set of tools for validating whether simulation projects and tools are consistent with these conventions; a collection of standardized Python APIs, command-line interfaces, and Docker images for simulation tools and the capabilities of these tools; a Docker image repository for these tools and a database for their specifications; a REST API for updating and querying this image repository and database; and a graphical user interface for browsing the database, validating projects, and getting recommendations for algorithms and tools.

The interfaces for simulation tools and the tools for validating simulation projects and simulation tools were primarily implemented with Python using community libraries such as jLibSED-ML, libCellML, libCOMBINE, libOmexMeta, libSBML, libSED-ML, pyBioNetGen, pyLEMS, pyNeuroML, RBApy, Smoldyn, and XPP. The containerized interfaces for simulation tools were developed using Docker. The tools for validating logs of the execution of simulation projects and the specifications of simulation tools, the database of simulation tools, the REST API to the database, and the web application were implemented in TypeScript using NestJS, MongoDB, and Angular.

The database, API, web application, and test suite for simulation tools are deployed using Mongo Atlas, Google Cloud, Netlify, and GitHub issues and actions, respectively. The containerized simulation tools are stored using GitHub Container Registry.

More information about the architecture, implementation, and deployment of BioSimulators is available in Section S6.

USE CASES

Sharing, reproducing, and reusing simulations

We believe that BioSimulators makes it easier to share, reproduce, and reuse simulations by providing consistent workflows for installing and running simulation tools. Once an investigator has learned BioSimulators, they can run a broad range of simulations involving a broad range of tools. For example, we believe that simple web applications for using BioSimulators to execute simulations, such as runBioSimulations (15), will empower peer reviewers to review simulations more deeply, leading to more predictive models.

Quality-controlling simulations

We believe that BioSimulators' tool for integrated validation of simulation projects is excellent for identifying problems and other potential issues with simulations. For example, we are working with multiple model repositories to identify and correct issues in published simulation projects. More information is available in Section S7.

Comparing simulation tools

BioSimulators' registry of simulation tools is ideal for comparing and testing tools. In particular, by comparing the outputs of multiple tools, BioSimulators could help identify potential errors in tools. For example, we have used BioSimulators to find and fix bugs in VCell.

Multiscale simulation with multiple algorithms and tools

By providing consistent Python APIs for simulation tools, we believe that BioSimulators makes it easier to combine multiple simulations of various subsystems and scales into multiscale simulations. In particular, BioSimulators makes it easier to combine simulations that require multiple model formats, simulation algorithms, and simulation tools. For example, the Vivarium Collective (16) has begun to develop capabilities for co-simulating multiple BioSimulators tools.

DISCUSSION

As highlighted above, BioSimulators simplifies simulation by making it easier to find, obtain, install and run appropriate tools for particular simulations. Importantly, BioSimulators supports a broad range of simulations and simulation tools by using community formats and ontologies to encapsulate and abstract the details of individual formats and tools, including model formats such as BNGL, CellML, and SBML; SED-ML; KiSAO; the COMBINE archive format; HDF5; and Docker. We anticipate that these capabilities will enhance several stages of the modeling life cycle. For example, we anticipate BioSimulators will encourage more reuse of published simulations by making it easier to execute simulations, spur multiscale simulations by making it easier to combine multiple simulations of individual subsystems and scales, promote more predictive simulations by empowering peer reviewers to more deeply review simulations, and stimulate higher quality simulation repositories by enabling more holistic validation of simulation projects.

Systemizing additional simulation domains

Going forward, we aim to work with the community to expand the BioSimulators ecosystem to additional domains, including adding additional formats, frameworks, and algorithms to EDAM, SBO, and KiSAO; developing conventions for using SED-ML with additional model formats; incorporating additional model formats into our simulation project validation suite; curating additional example COMBINE archives for our simulation tool test site; and developing interfaces to additional simulation tools. Currently, we are working with the CoLoMoTo community to expand BioSimulators' capabilities for logical modeling, such as simulation of trap spaces.

Accelerating more holistic simulation workflows

By building on SED-ML L1V3, BioSimulators is currently limited to simple simulation workflows that consist of models, modification of models, the simulation of models, basic calculations of simulation results, exporting simulation results, and 2D line and 3D surface plots. In contrast, real-world simulation studies often involve additional tasks, such as aggregating, normalizing, and integrating data from multiple sources; using this data to build and calibrate models; performing complex data reductions on simulation results; and generating a variety of visualizations of simulation results. Going forward, we aim to work with the community to develop a new version of SED-ML, which can capture a broader range of tasks, and to develop a workflow engine that can use multiple containerized tools to execute the individual tasks of these workflows modularly. This design would also make it easier for software developers to participate in BioSimulators by lowering the responsibilities of tools from executing entire workflows to executing individual tasks.

Enhanced recommendations of simulation methods

Finally, we also aim to develop an additional wizard that helps novices identify appropriate formats, frameworks, algorithms, and tools for their work. Our current recommendation services require users to have advanced knowledge of simulation methodology. In contrast, we aim to develop a wizard that asks users questions about the systems and scales they would like to model and recommends appropriate formats, frameworks, algorithms, and tools. We anticipate that this would help more investigators model biology.

AVAILABILITY

BioSimulators is freely available without registration at https://biosimulators.org. This website contains links to the simulation tools, REST API, examples, and documentation. The source code for BioSimulators is openly available under the MIT license. More information is available in Section S9.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

FUNDING

This work was supported by National Institutes of Health awards P41EB023912 and R35GM119771.

Conflict of interest statement.

None declared.

REFERENCES

1. Carrera,J. and Covert,M. W. (2015) Why build whole-cell models? *Trends Cell Biol.,* **25**, 719–722.
2. Marucci,L., Barberis,M., Karr,J., Ray,O., Race,P. R., de Souza Andrade,M., Grierson,C., Hoffmann,S. A., Landon,S., Rech,E. et al. (2020) Computer-aided whole-cell design: taking a holistic approach by integrating synthetic with systems biology. *Front. Bioeng. Biotechnol.,* p. 942.
3. Szigeti,B. et al. (2018) A blueprint for human whole-cell modeling. *Curr. Opin. Syst. Biol.,* **7**, 8–15.
4. Waltemath,D., Karr,J. R., Bergmann,F. T., Chelliah,V., Hucka,M., Krantz,M., Liebermeister,W., Mendes,P., Myers,C. J., Pir,P. et al. (2016) Toward community standards and software for whole-cell modeling. *IEEE Trans. Biomed. Eng.,* **63**, 2007–2014.
5. Ebrahim,A., Lerman,J. A., Palsson,B. O. and Hyduke,D. R. (2013) COBRApy: constraints-based reconstruction and analysis for Python. *BMC Syst. Biol.,* **7**, 1–6.
6. Bergmann,F. T., Hoops,S., Klahn,B., Kummer,U., Mendes,P., Pahle,J. and Sahle,S. (2017) COPASI and its applications in biotechnology. *J. Biotechnol.,* **261**, 215–220.
7. Clerx,M., Cooling,M. T., Cooper,J., Garny,A., Moyle,K., Nickerson,D. P., Nielsen,P. M. and Sorby,H. (2020) CellML 2.0. *J. Integr. Bioinform.,* **17**, 20200021.
8. Keating,S. M., Waltemath,D., König,M., Zhang,F., Dräger,A., Chaouiya,C., Bergmann,F. T., Finney,A., Gillespie,C. S., Helikar,T. et al. (2020) SBML Level 3: an extensible format for the exchange and reuse of biological models. *Mol. Syst. Biol.,* **16**, e9110.
9. Malik-Sheriff,R. S., Glont,M., Nguyen,T. V., Tiwari,K., Roberts,M. G., Xavier,A., Vu,M. T., Men,J., Maire,M., Kananathan,S. et al. (2020) BioModels—15 years of sharing computational models in life science. *Nucleic Acids Res.,* **48**, D407–D415.
10. McDougal,R. A., Morse,T. M., Carnevale,T., Marenco,L., Wang,R., Migliore,M., Miller,P. L., Shepherd,G. M. and Hines,M. L. (2017) Twenty years of ModelDB and beyond: building essential modeling tools for the future of neuroscience. *J. Comput. Neurosci.,* **42**, 1–10.
11. Smith,L. P., Bergmann,F. T., Garny,A., Helikar,T., Karr,J., Nickerson,D., Sauro,H., Waltemath,D. and König,M. (2021) The Simulation Experiment Description Markup Language (SED-ML): language specification for Level 1 Version 4. *J. Integr. Bioinform.,* **18**, 20210021.
12. Bergmann,F. T. et al. (2014) COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project. *BMC Bioinformatics,* **15**, 1–9.
13. Courtot,M., Juty,N., Knüpfer,C., Waltemath,D., Zhukova,A., Dräger,A., Dumontier,M., Finney,A., Golebiewski,M., Hastings,J. et al. (2011) Controlled vocabularies and semantics in systems biology. *Mol. Syst. Biol.,* **7**, 543.
14. Peters,M., Eicher,J. J., van Niekerk,D. D., Waltemath,D. and Snoep,J. L. (2017) The JWS Online simulation database. *Bioinformatics,* **33**, 1589–1590.
15. Shaikh,B., Marupilla,G., Wilson,M., Blinov,M. L., Moraru,I. I. and Karr,J. R. (2021) RunBioSimulations: an extensible web application that simulates a wide range of computational modeling frameworks, algorithms, and formats. *Nucleic Acids Res.,* **49**, W597–W602.
16. Agmon,E. and Spangler,R. K. (2020) A multi-scale approach to modeling E. coli chemotaxis. *Entropy,* **22**, 1101.