Nucleic Acids Research, 2022 **1**
https://doi.org/10.1093/nar/gkac331

BioSimulators: a central registry of simulation engines and services for recommending specific tools

Bilal Shaikh [1], Lucian P. Smith [2], Dan Vasilescu[3], Gnaneswara Marupilla[3], Michael Wilson [3], Eran Agmon [4], Henry Agnew [5], Steven S. Andrews [2], Azraf Anwar[6], Moritz E. Beber [7], Frank T. Bergmann [8], David Brooks [9], Lutz Brusch [10], Laurence Calzone [11], Kiri Choi [12], Joshua Cooper[13], John Detloff[14], Brian Drawert [13], Michel Dumontier [15], G. Bard Ermentrout [16], James R. Faeder [16], Andrew P. Freiburger [17], Fabian Fröhlich [18], Akira Funahashi [19], Alan Garny [9], John H. Gennari [20], Padraig Gleeson [21], Anne Goelzer [22], Zachary Haiman [23], Jan Hasenauer [24], Joseph L. Hellerstein [2], Henning Hermjakob [25], Stefan Hoops [26], Jon C. Ison [27], Diego Jahn [10], Henry V. Jakubowski [28], Ryann Jordan[1], Matúš Kalaš [29], Matthias König [30], Wolfram Liebermeister [22], Rahuman S. Malik Sheriff [25], Synchon Mandal [31], Robert McDougal [32], J. Kyle Medley[33], Pedro Mendes[3], Robert Müller [10], Chris J. Myers [34], Aurelien Naldi [35], Tung V.N. Nguyen [25], David P. Nickerson [9], Brett G. Olivier [36], Drashti Patoliya[37], Loïc Paulevé [38], Linda R. Petzold [39], Ankita Priya[40], Anand K. Rampadarath [9], Johann M. Rohwer [41], Ali S. Saglam [16], Dilawar Singh [42], Ankur Sinha [43], Jacky Snoep [41], Hugh Sorby [9], Ryan Spangler [44], Jörn Starruß [10], Payton J. Thomas [45], David van Niekerk[41], Daniel Weindl [46], Fengkai Zhang [47], Anna Zhukova [48], Arthur P. Goldberg [1], James C. Schaff [3,49], 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, USA, [2]University of Washington, Seattle, WA 98105, USA, [3]University of Connecticut School of Medicine, Farmington, CT 06030, USA, [4]Stanford University, Stanford, CA 94305, USA, [5]LibreTexts, USA, [6]New York University, Brooklyn, NY 11201, USA, [7]Unseen Bio ApS, 2100 København Ø, Denmark, [8]Heidelberg University, 69120 Heidelberg, Germany, [9]University of Auckland, 1010 Auckland, New Zealand, [10]Technical University of Dresden, 01187 Dresden, Germany, [11]Institut Curie, 75248 Paris, France, [12]Korea Institute for Advanced Study, 02455 Seoul, South Korea, [13]University of North Carolina, Asheville, Ashville, NC 28804, USA, [14]Independent, Madison, WI 53705, USA, [15]Maastricht University, 6200 Maastricht, Netherlands, [16]University of Pittsburgh, Pittsburgh, PA 15260, USA, [17]University of Victoria, Victoria, BC V8P 5C2, Canada, [18]Harvard Medical School, Boston, MA 02115, USA, [19]Keio University, Yokohama 223-8522, Japan, [20]University of Washington, Seattle WA 98019, USA, [21]University College London, London WC1E 6BT, UK, [22]Université Paris-Saclay, INRAE, MaIAGE, 78350 Jouy-en-Josas, France, [23]University of California, San Diego, La Jolla, CA 92093, USA, [24]Universität Bonn, 53115 Bonn, Germany, [25]European Molecular Biology Laboratory - European Bioinformatics Institute, Hinxton, Cambridge CB10 1SD, UK, [26]University of Virginia, Charlottesville, VA 22904, USA, [27]CNRS, UMS 3601, Institut Français de Bioinformatique, IFB-core, 91000 Évry-Courcouronnes, France, [28]College of Saint Benedict and Saint John's University, St. Joseph, MN 56374, USA, [29]University of Bergen, 5020 Bergen, Norway, [30]Humboldt University of Berlin, 10115 Berlin, Germany, [31]Technical University of Dresden, 01069 Dresden, Germany, [32]Yale University, New Haven, CT 06511, USA, [33]Autodesk, Inc., San Rafael, CA 94903, USA, [34]University of Colorado at Boulder, Boulder CO, 80309, USA, [35]Inria Saclay - Île-de-France Research Centre, 91120 Palaiseau, France, [36]Vrije Universiteit Amsterdam, 1081 HZ Amsterdam, Netherlands, [37]Sarvajanik College of Engineering &

[*]To whom correspondence should be addressed. Email: jonrkarr@gmail.com

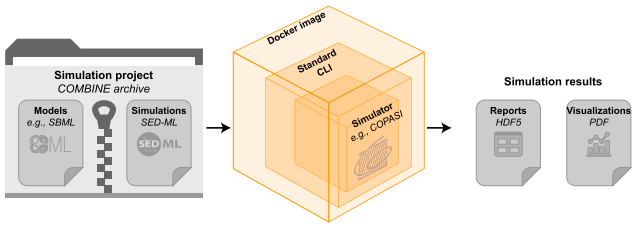
Technology, Surat, Gujarat 395001, India, [38]Centre National de la Recherche Scientifique, 33400 Talence, France, [39]University of California, Santa Barbara, Santa Barbara, CA 93106, USA, [40]Birla Institute of Technology, Mesra, Jharkhand 835215, India, [41]Stellenbosch University, Stellenbosch, 7600, South Africa, [42]Subconscious Compute Pvt. Ltd., Bangalore, India, [43]University College London, London, WC1E 6BT, UK, [44]Allen Institute for Cell Science, Seattle, WA 98109, USA, [45]University of Utah, Salt Lake City, UT 84112, USA, [46]Helmholtz Zentrum München GmbH and German Research Center for Environmental Health, 85764 Neuherberg, Germany, [47]National Institutes of Health, Bethesda, MD 20892, USA, [48]Institut Pasteur, 75015 Paris, France and [49]Applied BioMath LLC, Concord, MA 01742, USA

Received March 13, 2022; Revised April 07, 2022; Editorial Decision April 16, 2022; Accepted April 20, 2022

ABSTRACT

Computational models have great potential to accelerate bioscience, bioengineering, and medicine. However, it remains challenging to reproduce and reuse simulations, in part, because the numerous formats and methods for simulating various subsystems and scales remain siloed by different software tools. For example, each tool must be executed through a distinct interface. To help investigators find and use simulation 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 each version of each tool. The foundation of BioSimulators is standards, such as CellML, SBML, SED-ML and the COMBINE archive format, and validation tools for simulation projects and simulation tools that ensure these standards are used consistently. To help modelers find tools for particular projects, we have also used the registry to develop recommendation services. We anticipate that BioSimulators will help modelers exchange, reproduce, and combine simulations.

GRAPHICAL ABSTRACT



INTRODUCTION

Sophisticated computational models that can predict biological phenomena have great potential for bioscience, bioengineering, and medicine. For example, whole-cell 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 and the use of multiple tools (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 libCellML and libSBML; and repositories such as BioModels (9) and ModelDB (10) help investigators share and reuse diverse 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.

Despite this progress, sharing, reusing, and combining simulations remains difficult. One reason is that it is difficult to find, obtain, and use appropriate tools for particular systems and scales. For example, many tools do not provide clear documentation about their simulation methods, and each tool must be obtained from a different location, installed via a different process, and executed via a different UI or API using different model formats. Guides, such as the retired SBML Software Guide, and container registries, such as BioContainers, have only addressed some of these issues.

To accelerate the reuse of simulations, as well as 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 54 tools for 13 model formats, 14 modeling frameworks, and 91 simulation algorithms (Supplementary Tables S1–S3), and consistent interfaces to 21 of these tools (Supplementary Tables S4 and S5). For example, this includes asynchronous logical simulation with BoolNet, geometric flux balance analysis with COBRApy, discrete particle-based simulation with BioNet-

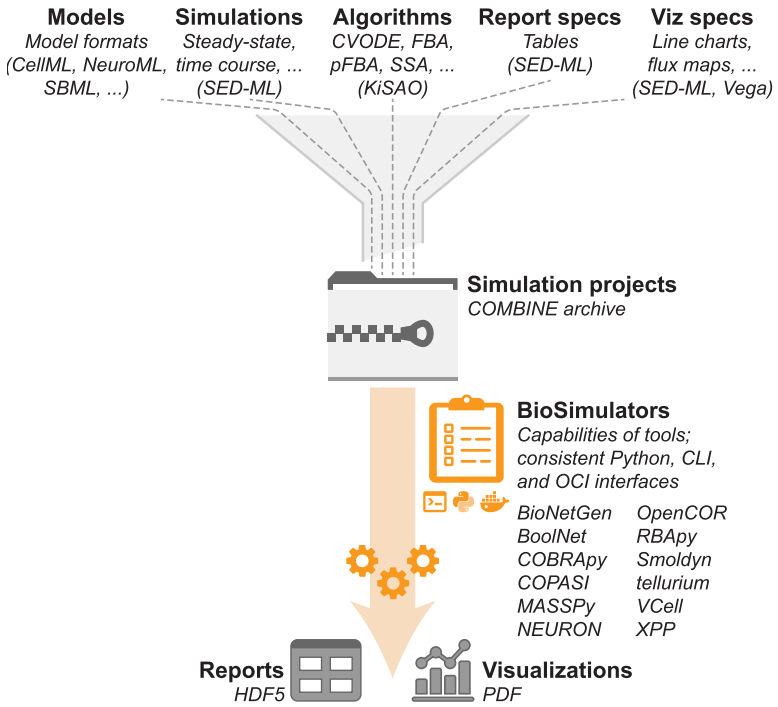


Figure 1. BioSimulators simplifies simulation by abstracting simulation projects and simulation tools. BioSimulators abstracts projects as COMBINE archives and tools as containerized command-line interfaces. These abstractions make it easier to execute a broad range of simulations.

Gen, and discrete spatial simulation with Smoldyn. 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 Docker images 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 data sets and visualizations of simulation results. 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 for using BioSimulators to execute and share simulations and visualizations of their results (15) (Figure 2C).

Below, we summarize the key features of BioSimulators, describe its architecture, delineate several use cases for BioSimulators, and outline the future directions of BioSimulators. Tutorials and additional documentation are available at https://docs.biosimulations.org.

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 metadata, 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 algorithms captured by KiSAO to develop recommendation services. For example, we have developed a web form that can recommend simulators for executing particular projects.

Streamlined acquisition and installation of simulators

To make it easier to obtain and install simulation tools, BioSimulators saves a Docker 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, which generally disallow the use of Docker due to security limitations, BioSimulators tests that these images are compatible with the Singularity Image Format (SIF), 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.

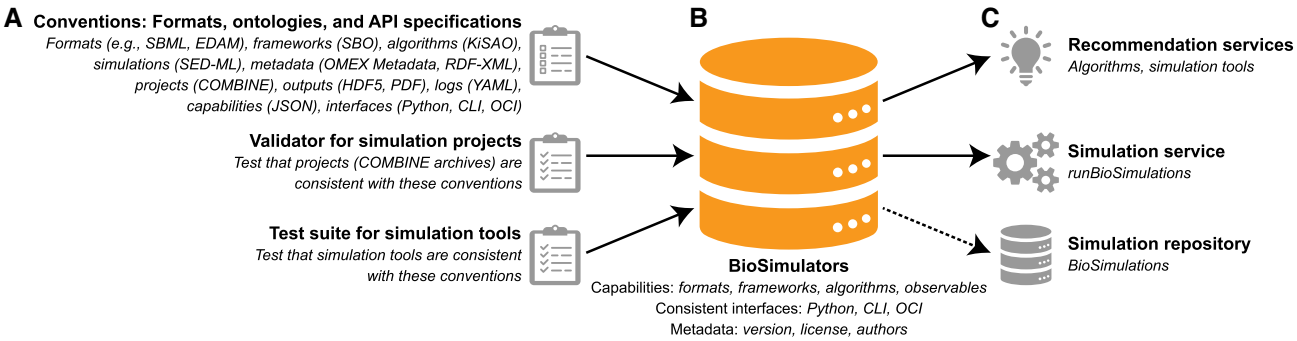


Figure 2. Overview of the BioSimulators ecosystem. The foundation of BioSimulators (**B**) is an integrated set of formats, ontologies, and specifications for simulation projects and simulation tools, and tools for checking that these conventions are used consistently (**A**). These conventions make 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**).

Streamlined execution of simulation tools

To simplify simulation, each containerized tool provides the same command-line interface. These interfaces capture the project to be executed and the location where its outputs (reports and visualizations) should be saved. This enables modelers to use multiple simulators simply by switching 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 simulators 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 simulators are consistent and that their specifications are accurate, we extensively review each version of each tool. The first version of each tool submitted to BioSimulators is both automatically validated by a test suite that we developed and manually reviewed by our team. Each subsequent version of each tool is also validated by this test suite. This test suite uses the simulator to execute a set of example COMBINE archives and checks that the tool produces the expected results. The test suite uses the specifications for the tool to select appropriate example archives. To enable developers to keep BioSimulators up to date, we provide an API that developers can use to automatically submit each version of their tools. We anticipate that this approach will enable us to keep BioSimulators up-to-date and accurate with minimal effort.

METHODS

Consistent representation of simulation projects and tools

The foundation of BioSimulators is a set of formats, ontologies, and specifications for consistent representation of simulation projects (one or more simulations of one or more models using one or more algorithms), the inputs (e.g. experimental data for validating simulations) and outputs (data sets and visualizations of simulation results) of simulation projects, simulation tools and the capabilities of simulation tools (supported formats, frameworks and algorithms) (Supplementary Figures S1 and S2). Where possible, these conventions embrace existing resources. Using these resources required filling numerous gaps within and between them. This included creating new schemas for simulation results, logs of the execution of simulations, and the capabilities of simulation tools; formalizing numerous aspects of SED-ML; adding many new ontology concepts for formats and algorithms; and correcting hundreds of bugs in various simulation projects and software tools.

BioSimulators uses the COMBINE archive format to encapsulate all of the files that constitute a simulation project. Within COMBINE archives, BioSimulators uses formats such as BNGL, CellML, GINML, NeuroML/LEMS, RBA XML, SBML, Smoldyn, VCML and the XPP ODE format 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. We use the PDF format to capture visualizations of simulation results.

To enable modelers to execute simulators consistently, we developed specifications for Python APIs and containerized command-line programs for simulators. 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 better capture the capabilities of a broader 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 wrapping 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 (i) resolving the model for the task, (ii) modifying the model according to the changes specified in SED-ML, (iii) using KiSAO to determine the most similar simulation algorithm that the simulation tool implements to the algorithm specified for the task, (iv) translating this algorithm and its specified parameters into the corresponding method of the simulation tool and its arguments, (v) executing this method with these arguments, (vi) collecting the results of this method and (vii) 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 use SED-ML to precisely record the algorithms they used to execute simulations with one tool and 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 our registry of tools; (b) a web form for obtaining a list of tools which implement algorithms similar to a given algorithm, sorted by the maximal similarity of their algorithms to the given algorithm and (c) a web form for identifying simulators which can execute a given project using the specified or similar algorithms. Briefly, we implemented these services by (a) determining the formats and algorithms specified for a given project, (b) using our registry to determine the capabilities of each tool, (c) using parent-child and other relationships to encode similarities among algorithms into KiSAO, (d) using these relationships to query KiSAO for sets of similar algorithms, (e) manually assigning each set a degree of similarity, (f) combining the formats and algorithms required for a given project, the capabilities of each tool, and the similarity among algorithms to determine the maximal degree of similarity at which each tool can execute a given project and (g) sorting the tools by this maximal similarity. 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 metadata files) and tools for validating simulation results, logs of the execution of simulations, and the capabilities of simulators described with the new schemas outlined above. 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 model variable. To make these validation tools easy to use, we developed several interfaces, including web forms, a REST API, a command-line program, and a Python API. Four model repositories are already using these tools to debug their models and simulations.

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 formats and algorithms, the test suite uses the specifications of tools to select appropriate archives for their validation from a corpus of curated archives and then uses these curated archives to computationally generate additional archives for testing specific aspects of BioSimulators' conventions. This design enables us to pinpoint issues with simulation tools, and it makes it easy to expand the test suite to additional model formats and methods. The test suite can be executed through a command-line interface or the GitHub issues deployment described below. More information about these validation tools is available in Section S3.

Submission of simulation tools to the registry

Developers can submit tools to the registry by submitting issues to the BioSimulators GitHub repository. Once an issue is created, GitHub actions is then used to execute the test suite described above, and any test failures are reported as messages to the issue. The first time a simulation tool passes the test suite, our team also manually reviews the capabilities of the tool and uses the issue to discuss any suggested revisions with the submitter. 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 rigorously review each version of each tool with minimal effort.

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 their tools up-to-date in BioSimulators. For example, developers can use this API within GitHub actions. Currently, half of the containerized tools registered with BioSimulators automatically release each version to BioSimulators. Third, GitHub issues enables our team to monitor problems that developers are encountering and help them.

DESIGN, IMPLEMENTATION AND DEPLOYMENT

BioSimulators is composed of a set of conventions for consistently representing simulation projects and simula-

tion 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; a Docker image repository for these tools; 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 (Supplementary Figure S3).

The interfaces for simulators and the tools for validating simulation projects and simulators were primarily implemented with Python using libraries such as jlibSEDML, libCellML, libCOMBINE, libOmexMeta, libSBML, libSEDML, pyBioNetGen, pyNeuroML, RBApy, Smoldyn and XPP. The containerized interfaces for simulators 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, 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 simplifying the installation and execution of simulators. Once an investigator has learned BioSimulators' conventions, they can run a broad range of simulations involving a variety of tools. In particular, we believe that simple web applications for using BioSimulators, such as runBioSimulations (15), will empower peer reviewers to review simulations more deeply, leading to better evaluation of modeling studies.

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, BioSimulators has helped the BioNetGen, pyNeuroML, VCell and other teams find and fix bugs in their tools.

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

In summary, BioSimulators simplifies simulation by making it easier to find, obtain, and run appropriate tools for particular projects. Importantly, BioSimulators supports a broad range of simulation projects by using several formats and ontologies to encapsulate and abstract individual formats and tools, including model formats such as BNGL, SED-ML, KiSAO, the COMBINE archive format, HDF5 and Docker. We anticipate that BioSimulators will enhance several stages of the modeling life cycle. For example, we anticipate BioSimulators will encourage more reuse of published simulations by simplifying their execution, spur multiscale simulation by making it easier to combine multiple simulations of various subsystems, promote more predictive simulations by empowering peer reviewers to deeply review simulations, and stimulate higher quality simulation repositories by enabling more holistic validation of simulations. Below, we summarize how we plan to continue to enhance BioSimulators.

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 suite; and developing interfaces to additional simulation tools. Currently, we are working with the CoLoMoTo community to expand BioSimulators' capabilities for logical modeling, such as calculations of state transition graphs and trap spaces.

Accelerating more holistic simulation workflows

By building on SED-ML, 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 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 develop a workflow engine that can use multiple containerized tools to modularly execute the individual tasks of these workflows. 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. This wizard will ask users questions about the systems and scales they would like to model and recommend appropriate formats, frameworks, algorithms, and tools. We anticipate that this would help more investigators model biology.

DATA 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

National Institutes of Health [P41EB023912, R24GM1 37787, R35GM119771]. Funding for open access charge: NIBIB [P41EB023912].
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.*, **8**, 942.
3. Szigeti,B., Roth,Y.D., Sekar,J.A.P., Goldberg,A.P., Pochiraju,S.C. and Karr,J.R. (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**, 74.
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., Adams,R., Moodie,S., Cooper,J., Glont,M., Golebiewski,M., Hucka,M., Laibe,C., Miller,A.K., Nickerson,D.P. *et al.* (2014) COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project. *BMC Bioinformatics*, **15**, 369.
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., Spangler,R.K., Skalnik,C.J., Poole,W., Peirce,S.M., Morrison,J.H. and Covert,M.W. (2022) Vivarium: an interface and engine for integrative multiscale modeling in computational biology. *Bioinformatics*, **38**, 1972–1979.