Briefings in Bioinformatics, 22(3), 2021, 1–13

doi: 10.1093/bib/bbaa152
Method Review

Publishing reproducible dynamic kinetic models

Veronica Porubsky, Lucian Smith and Herbert M. Sauro

Corresponding author: Herbert M. Sauro, Department of Bioengineering, University of Washington, Seattle, WA 98105, USA. Tel: Telephone: +206 685 2119; E-mail: hsauro@uw.edu

Abstract

Publishing repeatable and reproducible computational models is a crucial aspect of the scientific method in computational biology and one that is often forgotten in the rush to publish. The pressures of academic life and the lack of any reward system at institutions, granting agencies and journals means that publishing reproducible science is often either non-existent or, at best, presented in the form of an incomplete description. In the article, we will focus on repeatability and reproducibility in the systems biology field where a great many published models cannot be reproduced and in many cases even repeated. This review describes the current landscape of software tooling, model repositories, model standards and best practices for publishing repeatable and reproducible kinetic models. The review also discusses possible future remedies including working more closely with journals to help reviewers and editors ensure that published kinetic models are at minimum, repeatable.
Contact: hsauro@uw.edu

Key words: systems biology; kinetic modeling; SBML; standards; reproducibility

Introduction

The use of simulation and theory in cellular and molecular biology has a long, if uneven, tradition spanning almost 90 years [115]. In the past 2 decades, however, there has been a significant increase in the use of mathematical approaches to help understand and predict cellular behavior. The total number of published biochemical models is not known, but we suspect it is on the order of 5 000, perhaps more. Unfortunately, the vast majority of these models are either not accessible or are very difficult to obtain. When publishing an experimental paper, there is often an extensive methods section that gives a detailed account of the methodology used to conduct the experiments to ensure that the reported results can be reproduced by a third party. Unfortunately, method sections for computational studies that explain how a model was built, simulated and analyzed are often absent or incomplete. In recent years, there have been substantial efforts to change this culture through the development of model exchange formats and, most importantly, model repositories. In this article, we will describe the current state of reproducibility in computational models with a focus on cellular pathway kinetic models. The article will not cover constraint-based metabolic modeling to any significant degree, as this topic is reviewed elsewhere [35, 44].

Recent efforts have been perusing the need that data should be more accessible. The acronym, FAIR (findable, accessible, interoperable and re-usable), is a rallying cry for a European initiative, (https://fair-dom.org/), to improve data management in the life sciences with a particular emphasis on accessibility

Veronica Porubsky is a graduate student who is working on building neuro-physiology mechanistic models. She is also head of dissemination at the NIH Center for Reproducible Biomedical Modeling.
Lucian Smith is a research scientist working at the Sauro laboratory. He has 15 years of experience in computational biology. He is currently working on developing standards that will help define how a model is simulated and analysed.
Herbert M. Sauro has worked as an academia in systems biology and synthetic biology for over 20 years in the US and before that for an additional 15 years in the UK. His main areas of research include metabolic control analysis, software infrastructure for modeling, and standards such as SBML. He is currently director of the NIH Center for Reproducible Biomedical Modeling.
Organization The University of Washington is a public university in Seattle. It was founded in 1861 and is one of the oldest universities on the West Coast of the USA. The university has approximately 32 000 undergraduates and 12 000 graduates.
Submitted: 7 February 2020; **Received (in revised form):** 19 May 2020

and reuse of scientific data [109]. The National Institutes of Health (NIH) in the USA has also recently voiced approval of FAIR and its use in NIH funded efforts (https://datascience.nih.gov/data-ecosystem).

Repeatability and reproducibility

Perhaps one of the most important elements of science is the requirement that observations should be independently reproducible. That is, another researcher, unconnected with the original study, should be able to carry out the same experiments and obtain the same results within some statistical tolerance. Confidence in the likely truth of a hypothesis is determined by the number of observations carried out by independent researchers that are consistent with the hypothesis. Science may not be able to reach the exact truth, but it can approach the truth through repeated observations and refinement of the hypothesis. In experimental science, this methodology has been in use intermittently for at least 2000 years (Philon of Byzantium, Pneumatica, 2nd century, BC) but has been implemented systematically in the last 1000 years starting with work by Ibn al-Haytham on optics in 1021 and, most famously, by Galileo Galilei on motion around 1590.

Science, since its beginnings, has also relied on quantitative measurement to buttress its conclusions. In more recent times, especially with the advent of electronic computers, quantitative measurement has taken on a much more significant role. Numerical calculations that support a hypothesis can now be very complex. Just as in experimental science where assumptions must be explicitly stated and procedures carefully described, the same care and attention should be applied to computational experiments; otherwise, it is not possible to reproduce the reported results. Two independent studies by the BioModels group at European Bioinformatics Institute (EBI) (https://www.ebi.ac.uk/biomodels-main/) and the Physiome Repository curators (https://models.physiomeproject.org/welcome) in Auckland, where over 1200 models were tested, concluded (personal communication) that 97% to 100% of all the tested computational models in systems biology and physiology were not reproducible.

Before proceeding, we should define what we mean by reproducibility (Figure 1). There is a surprising amount of disagreement in the scientific literature on the meaning of specific terms such a reproducible, repeatable and replicable [10, 90]. Here we will take a simpler approach and define two terms 'repeatability' and 'reproducibility' [34]. These terms have been used by the experimental biological communities [17], and their definitions closely match those used by the Association for Computing Machinery [6].

Repeatability

Repeatability is the ability of a researcher, using the same data and computational hardware and software, to repeat a study. That is, a third party researcher should have access to exactly the same source code and data that was used in the original study. In principle, the same operating system (Linux or Windows) and computer hardware (Intel or ARM-based computers) should also be used. Often the underlying hardware and operating system are relatively benign, but this is not always the case, as demonstrated by a recent and well-publicized instance in which the underlying operating system had a marked effect on calculating Nuclear Magnetic Resonance chemical shifts [16].

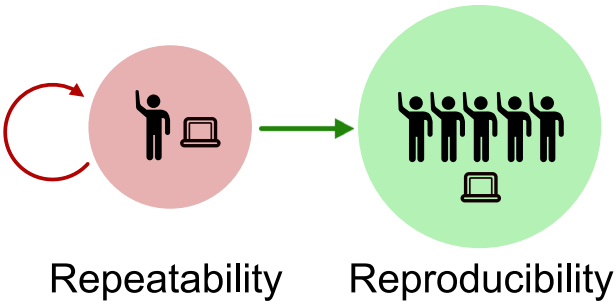

Figure 1. Repeatability and reproducibility. In this text, we classify repeatability as when the author of computational experiment provides all the necessary code to execute the computation. This can even take the form of a Docker image, for example. By reproducibility, we refer to the case when a third party recreates *de novo* some or all of the analysis described by a researcher.

Repeatability is the most basic test on the reliability of a computational experiment. In principle, it should be very easy for a researcher to pass their source code and data to a third party to repeat the calculations. However, it is surprising that even this simple and most basic requirement is often not met in published articles. One of the authors reviewed six papers in 2019 that included computational models and not a single model could be repeated as published. An interesting example of this problem can be found in the supercooled water community [7].

Reproducibility

Repeating a computation is not a strong statement about our confidence in or quality of the model but should be the minimum requirement when publishing a computational paper. The weakness in repeating a computational experiment is that one is also repeating any errors in the source code, input data or assumptions made by the original authors. To reproduce a computational experiment means that a researcher reconstitutes *de novo* one or more aspects of the study; the more aspects that are reconstituted, the better. At one extreme, a third party may re-implement from scratch the entire software workflow used by the study, using a completely different computer language, with the model and its assumptions revisited. They may also collect their own data or use alternative but, in principle, equivalent algorithms. It should be evident that there can be many degrees of reproducibility, perhaps only part of the software is rewritten, the same exact algorithms are used, etc. Reproducing a computational experiment and obtaining the same result is a much stronger statement than simply repeating an experiment. This is especially the case when the computational workflow is complex and relies on statistical analysis that include Monte Carlo sampling, where sample sizes, repeated runs and even the random number generator can influence the final results. For example, in kinetic modeling, it is now common to use Monte Carlo sampling of parameter values to generate ensembles of models [61, 108]. Reproducing such computational experiments is not always easy and depends heavily on the details provided by the original authors, such as the size of the ensemble or the assumed distribution for the sampled parameters.

What measures can be taken to improve the publication of repeatable and reproducible models, and why are so many models not reproducible? This is not a problem confined to the systems biology community but is endemic throughout science [9, 58].

Table 1. Simple definitions of terms versioning, validation and verification

Versioning	When a document, software, model, etc., goes through a number of drafts, such drafts are called versions, and the act of storing and keeping track of the versions together with a record of changes between drafts is called versioning.
Verification	In terms of a simulation model, verification is the process by which we determine that the model and numerical methods have been implemented correctly. For example, one might check that the differential equation solver has been implemented correctly or that the model equations were entered precisely. Verification does not check whether the scientific assumptions used to build the model are correct or whether predictions made by the simulation match experimental data.
Validation	Validation of a simulation model is where we check that the predictions made by the model match data gathered from the real system.

The reasons for failure to reproduce or even repeat a study are surprisingly mundane. They include missing values for parameters or initial conditions, errors in the model equations or units and errors in describing parameters (an author may refer to a radius when they intended to specify a diameter, for example). For repeatability, problems arise due to a failure to provide all the source code or even any of the source code. Other issues include the original simulation environment is unavailable, or the wrong model may be supplied with the paper.

This state of affairs most likely arises because the reward system for publishing reproducible or even repeatable computational experiments is absent. A recent report on reproducibility by the National Academy of Sciences [79] refers to these pressures as 'misaligned incentives'.

Given the many pressures on researchers and the way research is rewarded, work that does not contribute to future grant success or job promotion tends to receive less attention. Of course, some non-reproducible or repeatable studies can still have value [40]. However, there are more serious issues with the question of reproducibility. For one, there is a huge and therefore costly effort by thousands of researchers to get models working again in order to build on previous work. As models become more complex and begin to be used in medical or industrial domains, the quality of a model starts to become much more important. For example, if a model were to be used in a clinic to direct drug intervention, it is likely that extensive testing and validation would be demanded by government authorities to ensure that the predictions provided by the model can be safely used to treat patients. If models are more easily reproduced and especially if they use reproducibility standards to be described below, models will be more easily reused in other projects, more understandable if annotated and, most importantly, a testing, verification and validation frameworks could be used to ensure the model performs as expected (Figure 2). The terms versioning, verification and validation are often used in the literature but are rarely defined in simple terms. We have therefore provided simple definitions of these terms in Table 1.

In the following sections, we will describe efforts to make modeling more systematic and reproducible. We consider two key aspects: (i) standards to ensure consistency and transparency and (ii) a cultural shift in the way we report dynamic kinetic models of biochemical pathways.

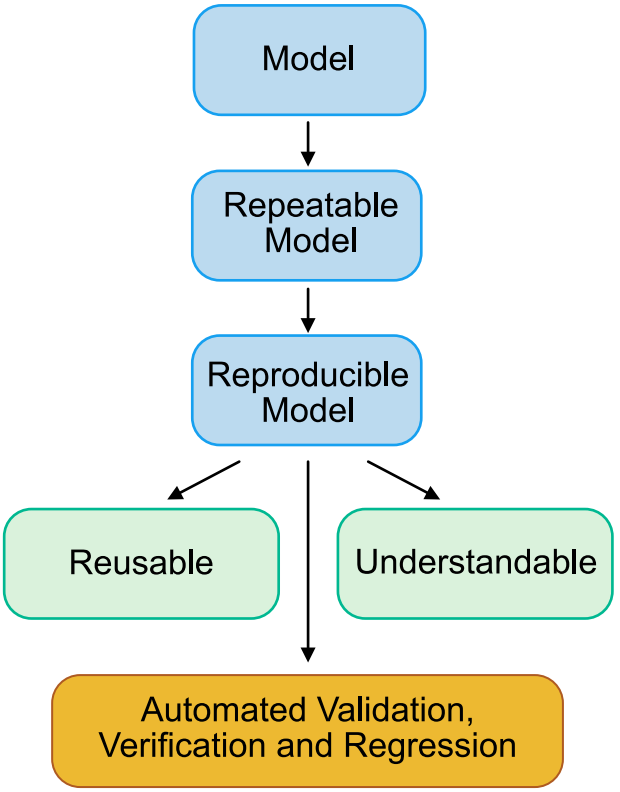

Figure 2. Importance of reproducible models. Reproducibility is important not only because it strengthens confidence in a particular result, but also it offers avenues to increase the understandability of a model, the reuse of a model and a host of additional opportunities that include automated testing, validation and verification of models.

Describing models

Published kinetic models are distributed in a variety of ways. These range from a description of the model in the appendix or supplement, simple text files, Excel spreadsheets, Matlab scripts, etc. Sometimes models are stored on the authors' web site or hidden behind a password-protected site. In these cases, the model will often disappear at a later date when the web site is updated or disbanded.

These formats are not interchangeable and are often not documented well. The reuse of such models can be very difficult, if not impossible, at times. The use of Matlab or other scripts to distribute kinetic models is particularly problematic because all the network information is lost or, at least, very difficult to recover. Another issue with *ad hoc* distribution methods is the lack of consistency in naming biochemical species or reactions in a model. This can make it very difficult to identify the components in the model. The solution to all these problems is to devise a standard format that is accepted by the modeling community.

The systems biology markup language

Over the past 2 decades, a variety of standards have been developed to assist with the exchange and formal representation of kinetic models. The most well known of these is the systems biology markup language (SBML) [56].

SBML was originally developed to meet an immediate need to allow researchers to exchange models between different modeling tools, in particular Gepasi [77], SCAMP [97] and Jarnac [95]. SBML represents models as a list of chemical transformations and employs specific elements to represent spatial compartments, molecular species, chemical transformations and parameters. In addition to these, SBML provides rules that can be used to represent constraints, derived values and general math, which, for one reason or another, cannot be transformed into a chemical scheme. SBML is an eXtensible Markup Language based format and, therefore, not very human readable. Instead, users typically use one of a number of SBML editors and viewers to interpret the model [2, 28, 66]. A readable form of SBML is the Antimony script language [103], which allows modelers to express models in an easily recognized reaction scheme format using a simple text editor. It supports the bulk of the SBML specification, including the ability to easily annotate models. Modeling tools such as Tellurium [25, 75] fully support the use of Antimony.

SBML is widely used by the pathway modeling community; most journals recommend that authors supply their models in SBML format. Given its structure, SBML is very well suited for representing pathway models and combined with Antimony, provides a very easy-to-use mechanism for building such models [91].

As the SBML community grew, there was a need to support additional information for particular modeling efforts. This was achieved by allowing SBML to be extended using packages [57]. For example, there is an extension specifically tailored towards metabolic constraint models (flux balance analysis) [35, 86]. Until the development of FBC, constraint-based models were often stored in Excel spreadsheets using notation that varied from one model to the next. This hindered the reproducibility of constraint-based models [35].

Other useful packages for the modeling community include the network layout and render package [15, 41], which allows a biological pathway model encoded in SBML to be visualized in a reproducible way. Hoksza *et al*. [54] offer a comprehensive review of this particular topic.

Annotation

Annotation of a model refers to adding metadata to a model description that will unambiguously identify components of a model. For example, a user might use the symbol 'X' to represent adenosine triphosphate in a model. Once published, it may be difficult to identify what molecule 'X' actually represented. Information that is included to indicate the identity of components or processes is called semantic annotation. Thus, alongside 'X' there would be a reference to an unambiguous term representing ATP. Such information [42] can be useful [106] for a number of reasons that include re-purposing models, deconstructing models in their identifiable component parts, composing larger models from smaller models and giving improved meaning to the parts of the model to indicate how the model was built, why, by whom and what assumptions and data sources were used in its construction. Finally, once models are annotated, they can be much more easily searched.

Annotations can be encoded in a variety of ways, but a common approach is to use Resource Description Framework (RDF). Online resources exist, such as the BioGateway [4] or BioPortal [93] that can be used to query information and resolve references. The use of model annotation is still in its infancy, but there is growing interest in using such resources, especially when constructing large models or libraries of submodels [29]. For genome-scale models, this is a particularly important problem given the many thousands of components in such models. Without a way to identify such components, comparing and curating such models becomes very difficult. Later we will discuss the COMBINE archive, which has specific support for RDF metadata storage.

Annotations can be added to SBML using a variety of tools. These include Antimony [103], which is distributed with Tellurium [25, 75], the graphical user interface (GUI) tool SemGen [80], a web-based annotation tool such as semanticsbml [72] and modeling tools such as COPASI [55] and JWS Online [85, 88]. Models could also be described using a simple pseudo-code, such an example can be found in simpleSBML [20] or SBML-shorthand [45]. For describing execution algorithms, a simplified version of Python could be used.

Modeling experiments

SBML is very useful for describing biochemical reaction models, with species, compartments, reactions and rate laws. However, it cannot be used to describe what computational experiments should be performed on the model. For reproducibility, this is of critical importance. For example, a paper may contain six figures describing different simulations or analyses performed on a single model. Even if the model itself is available, reproducing the published results can be difficult. As a result, there has been an effort to create standard ways in which computational experiments can be described. We will describe two such approaches, workflows and the simulation experiment description markup language (SED-ML) [14].

Workflows have become a popular means by which to describe a computational process and are widely used in bioinformatics studies [8, 68].

Workflow systems are suitable when computations are done using distributed resources, data sets and devices. This can occur in large scientific experiments such as those found in high-energy physics or genomic studies. Workflows are useful when tasks must be repeated often, tasks requiring distributed processing, possibly via remote resources and access to data distributed across the network. Often a workflow will call a variety of third-party software applications to accomplish the various stages of the computational experiment. There exist specialist tools such as Taverna [111], Galaxy [48] and Pegasus [30] that deal exclusively with managing workflows.

In the simplest case, a workflow can be expressed in a scripting language such as Python or MATLAB where all processing is done in-house and the scripts do not call external third-party applications. At the next level, scripts can call external third-party applications to accomplish some or all of its tasks. Finally, a researcher can use a dedicated workflow engine such as Galaxy. We thus see a range of approaches to implementing workflows, each with its advantages and disadvantages. One disadvantage is that workflow languages tend to be more coarse-grained in their descriptions of computational experiments. Thus, the detailed description of how to do a complex simulation study is not always possible. Such detail is often relegated to external applications using other description languages.

A particular concern is whether workflows can be reproduced long after the workflow was released. For example, myExperiment [47] is a hosting site for a large number of workflows. However, a casual inspection shows that a number of workflows hosted there can no longer be executed, either because the particular workflow engine is no longer supported, or the resources the workflow were dependent on are no longer available. This is a known issue called 'workflow decay' [52, 92], where, over time, the environment within which the workflow was designed to execute has changed, making it difficult or impossible to reproduce the original workflow. Such changes include revisions to the data sources, changes to the software applications the workflow uses, changes to the underlying operating system or changes to the workflow engine itself. All these can render a workflow unable to reproduce the original results. In some respects, workflow systems are less reliable than if authors simply distributed MATLAB or Python scripts. There are mechanisms to prevent workflow decay. For example, the user can containerize the entire workflow system together with data and any third-party tools using tools such as Docker [23]. We recommend distributing containerized workflows and providing good documentation to guide users through installation and usage.

An alternative to using a workflow is to use a dedicated domain language such as SED-ML. SED-ML [107] uses a declarative language expressed using XML. However, computational experiments are generally procedural in nature, and as a result, it can difficult for SED-ML to express the full richness of many simulation experiments. SED-ML supports the following operations: time course simulation, steady-state computation, iteration, numerical algorithm specification, a limited set of post-processing operations and outputs to specify how the results of a simulation should be presented. Because SED-ML has only limited capabilities and the generation of SED-ML is non-trivial for novices, the proposal has not achieved as much traction as it could have. There have been efforts to make it easier to use, for example, Bergmann [12] proposed a simple Python syntax for generating SED-ML scripts, a human-readable form of SED-ML was developed by Smith and Choi *et al*., called phraSED-ML [26] and a GUI tool by Adams for generating SED-ML workflows [1] is available. There have also been alternative proposals such as simulation experiment specification via a Scala layer [39].

Workflow systems and domain-specific languages such as SBML and SED-ML can be combined for the best of both worlds. Docker images can be built for a modeling workflow, such that the operating system and all software libraries, applications and other dependencies are prepared for distribution. To build the Docker image, a Dockerfile must be prepared that specifies any parent image it is derived from, sets a working directory, installs dependencies, copies relevant files into the image and runs a modeling application or workflow from executable programs. In the case of biochemical modeling workflows, relevant files would include all data used by the model, model descriptions (e.g. SBML file) and all simulation experiments (e.g. SED-ML files) described in published work. The simulator used to run the simulation experiments would be packaged as a software dependency (e.g. by including the command: pip install tellurium). An example Dockerfile is shown in Figure 3. Then, the command line in the Dockerfile will specify the programming language to use (e.g. Python) to run the given executable (e.g. a Python script that loads the SBML and SED-ML files, calls the simulator and generates simulation results). After the build is complete, all software associated with the workflow is fully contained within the image and is robust to changes or versioning of the dependencies. This Docker image can be pushed to Docker Hub for distribution, which then enables interested users to execute a single command line pull request for installation. This provides a simple installation method for end users, complete with the ability to request specific versions of the image. Once installed, running a Docker container from the image can be readily performed from the command line.

To see a containerized modeling workflow that incorporates standardized modeling formats, we encourage readers to explore the executable simulation model (EXSIMO) [67]. This is an exemplary study that describes how to create a reproducible model of the liver and it can serve as a practical guide for implementing Docker containers to distribute the model and its simulation studies. The EXSIMO platform has implemented an executable simulation model of the liver using the SBML model description format, and simulation experiments are described to be compatible with SED-ML.

Alternatively, Docker can be used to distribute simulations or applications enabling the model and its simulation studies to be manipulated and executed, through a web browser using a Linux-based Docker image. In this case, the Docker image used to render the application, including the model, simulations and graphical output of the results, is typically hidden from the end user. While the developer will need to have expertise in generating the web application, this method ensures that end users may readily interact with a familiar GUI from the browser rather than execute commands from the command line. This approach is accessible to modelers with a range of computational backgrounds and expertise, widening the pool of potential end users who wish to repeat published simulations.

Instead of promoting workflow systems, the systems biology modeling community has focused on specifying standards and developing tools to support them, knowing that user tools have a short shelf life. This means that even though one tool might cease to be maintained and even operate correctly, other tools will emerge to replace it. So long as tools adhere to the agreed upon community standards, users are not dependent on any one tool, meaning reproducibility is more likely to be assured over the long term. The downside to this approach is that it requires considerable engagement with the community in order for standards to gain acceptance.

For small-scale reproducible experiments, particularly the kind found on BioModels and other model repositories, it might be more robust and convenient to write computational experiments in an open scripting language such as Python or to encode the experiment using platform-independent standards such as SED-ML. Another recommendation is to transcribe all the equations, parameters values, etc. into an appendix of the paper. One has to be very careful to ensure that the transcribed model is properly curated before publication. For large models, it is very easy to introduce errors at this stage. One option is to create an automated system that converts the machine readable model into a more readable format so that there is no instance where a human could introduce errors. Tellurium, for example, provides a means to convert SBML into a set of human readable equations.

Packaging modeling studies

Given that a single model may encompass many different kinds of data and specifications, such as SBML for the model, SED-ML for the simulation description, data for input to the model, etc., an archival format was proposed called the COMBINE archive (or

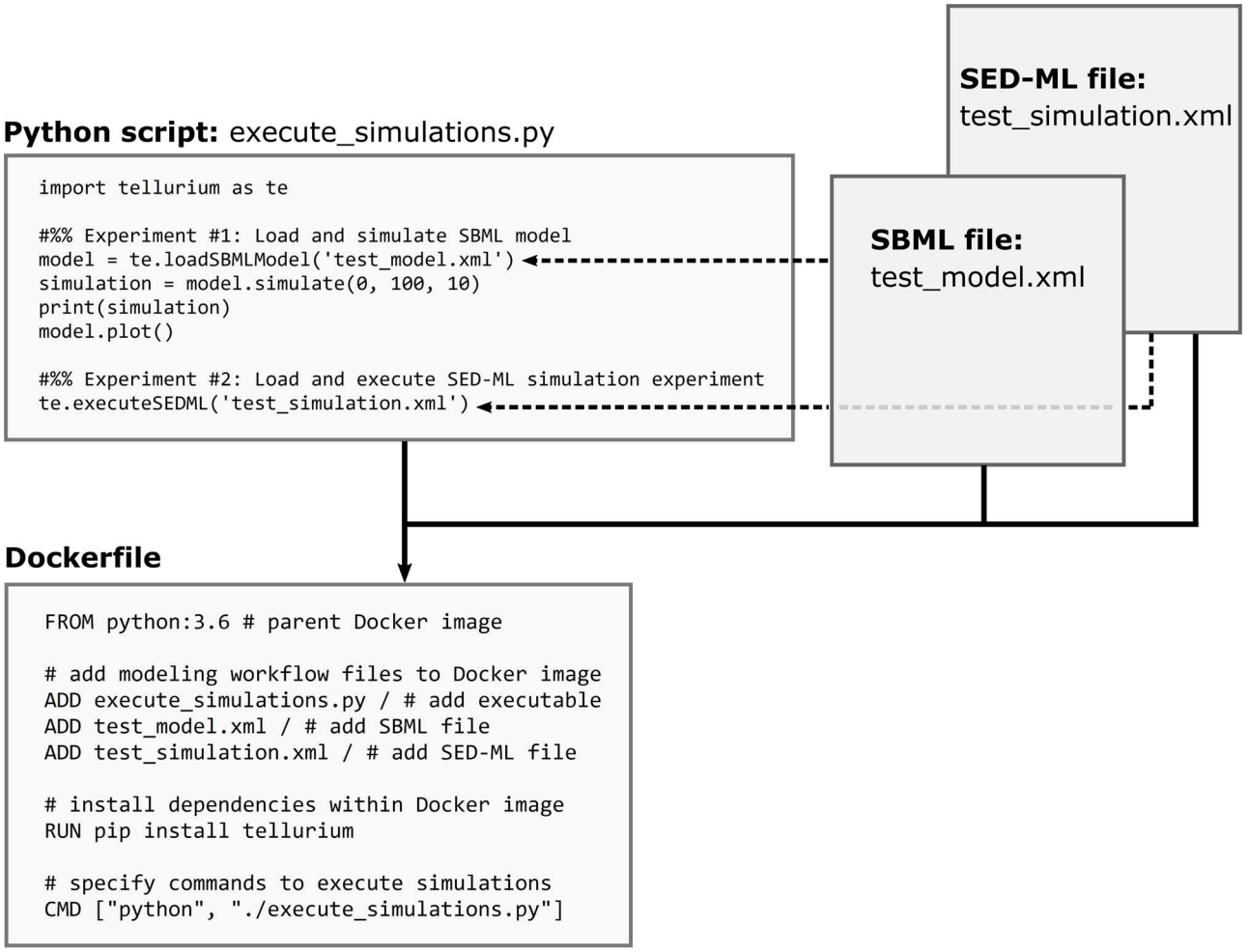

Figure 3. Example of generating a Dockerfile to distribute containerized models and simulation experiments composed of standardized formats (SBML and SED-ML).

OMEX file format) that would package all the necessary component files into one comprehensive file [13]. Much like a modern Microsoft Office Word file, a COMBINE archive is a zip archive that contains an XML-based manifest file. A ZIP file allows files to be stored hierarchically using folders, but in a compressed format, thus making it easy to transport. In addition to the modeling files, a COMBINE archive also contains an XML-based manifest file and an optional metadata file in RDF format [81]. The manifest can be used by software to determine the contents of the archive, and the metadata file can be used to identify unambiguously components found in the various modeling files. For example, a species named 'X' in an SBML model may be identified as glucose by the metadata file. A growing number of tools support the COMBINE archive, and in the future, model repositories are likely to allow users to download a complete description of a modeling study in the form of a single COMBINE archive.

Finally, the EXSIMO study mentioned previously demonstrates efficient packaging of a complete modeling exercise [67]. The approach packages the data, model and code necessary to repeat the studies (https://github.com/matthiaskoenig/exsimo). The workflow uses python to orchestrate all analyses using the EXSIMO python package. The model and associated data and simulation workflow can be stored on a GitHub repository. This allows model variants and simulations to be generated using GitHub's version control system. The approach at present requires some technical expertise but illustrates how a packaging systems might be implemented going forward. The example provided also illustrates how one might incorporate a set of tests into the workflow, akin to unit testing in software.

Software tooling

The systems biology has developed a rich suite of tooling to help with simulation, annotation and packaging. In this section, we will focus on software libraries that can be used to develop other software or which can be used from scripting languages such as Python.

Libsbml [18] is a software library provided by the SBML team that can be used to read and write SBML. The library is based around a C/C++ core, with wrappers provided for many programming languages. Furthermore, the library is available for Windows and POSIX operating systems and thus can be used virtually anywhere. With an abundance of documentation and available examples, software developers can readily use libsbml for their SBML support. By using libsbml, a developer can focus on how to interpret computational models rather than concern themselves with the mechanics of reading and writing models. Libsbml is a very well-developed library that sets a high standard for the development of other software libraries in the systems biology community. For web-based applications, a Javascript/Web Assembly version of libsbml is also available

[76]; a demonstration application that uses the web version of libsbml can be found at: https://libsbmljsdemo.github.io. For Java developers, a Java version of libsbml, called JSBML, is also available [33].

Along with libsbml, there are also additional libraries such as libsedml, libcombine and libantimony that can be used to read and write SED-ML files (https://github.com/fbergmann/libSEDML, see below), combine archives ([13], see below) and Antimony scripts ([103], see above).

LibRoadrunner [104] is a library that provides simulation support for SBML models. It offers both C++, C and Python application programming interfaces. It uses just-in-time compilation to achieve very fast simulation times. It is an order of magnitude faster and any other simulation package that uses SBML [74] and is equivalent in speed to a model encoded in raw C. For this reason, libRoadRunner is an ideal simulation library for doing high performance distributed parameter fitting where runtimes can be significantly reduced as a result [19, 78]. LibRoadrunner also links to other tools such as AUTO2000 [87] to provide bifurcation analysis of SBML models. For Java developers, the systems biology simulation core library is available [64] and can be used to build Java-based applications compatible with SBML.

Modeling software

Simulating a kinetic model involves integrating the equations whichthat describe the model to generate predictions of the dynamic behavior that the system will achieve under a given parameter regime and initial conditions. As models incorporate more equations, species and parameters, the computational costs associated with simulation increase. As a result, we focus on simulators that can perform rapid computation and provide additional analyses such as metabolic control analysis [63, 98] other than simple solutions to differential equations. More extensive reviews of software used in systems biology modeling can be found in [49, 96].

One area of software provision that we do not cover, but which is likely to have a major impact in the future, is static and dynamic testing of models. Probably the most well known of these is Memote [73]. Memote does static tests of an SBML model with a focus on genome-scale constraint-based models. However, one could imagine dynamic tests of models to verify that the software and check that the model is not violating physical laws. Metamorphic testing [101] is a method of testing that has found some application in bioinformatics [24] and could be used in systems biology modeling.

Tellurium

Tellurium [25] is a programming environment that provides a range of utilities to support kinetic modeling by combining languages for human-readable and human-writeable modeling (Antimony [103]) and simulation experiment descriptions (PhraSED-ML [27]) with a powerful underlying simulator (libRoadRunner [104]), while retaining flexibility by integrating any Python libraries desired for numerical or systems analysis. LibRoadRunner provides a simulator for SBML and is the highest performing SBML simulator by a wide margin [75] due to the use of a custom just-in-time compiler for SBML [104]. Performance also scales well for large models compared to alternative SBML simulators, as demonstrated in Maggioli *et al.* [75]. LibRoadRunner can handle deterministic and Gillespie-based stochastic time course simulations of kinetic models and supports steady-state analysis, stability and structural analysis of the stoichiometry matrix and metabolic control analysis [104]. The deterministic integrators are based on the CVODE integrator from the Sundials suite [53] and the fourth-order Runge–Kutta method, while the steady-state solver implements the NLEQ [83] Newton-based method. Tellurium supports SBML, SED-ML and COMBINE archives, improving exchangeability across simulation platforms [25]. Tellurium notebooks via Jupyter can also be used to support model reproducibility and reuse by providing an integrated environment in which SBML models, Python code and text-based descriptions of the model, code and simulation experiments can be displayed together for ease of comprehension and use [75].

Complex pathway simulator

The complex pathway simulator (COPASI) was developed to provide a complete suite of tools for simulation and analysis of biochemical reaction networks [55]. COPASI can be used as a GUI or through a command-line version if only numerical calculations are necessary [55]. The GUI version provides utilities for model editing and plotting simulation and analysis results [55]. The COPASI simulator supports numerical operations, including deterministic and stochastic numerical integration, steady-state calculation, stoichiometric and metabolic control analyses and computation of Lyapunov exponents. The LSODA integrator [89] is used for deterministic simulations, while the Gibson–Bruck Gillespie method [43], Gillespie's direct method [46], tau-Leap or adaptive SSA/tau-leap [21] is used for stochastic simulations [55]. A hybrid method is also available [55]. For steady-state computations, a damped Newton method with forward or backward integration is performed using LSODA [55]. While Tellurium interfaces with Python to provide flexibility for iterative programmatic tasks, the COPASI GUI also provides support for iterative tasks like parameter scanning and estimation and optimization [55]. COPASI also provides SBML support and import-export capabilities [55].

JWS online

JWS online and its simulation database were created to link simulation studies, specifically those described in SED-ML files or the COMBINE archive, with their associated results, models, model metadata and provenance information while enabling simulation experiments to be performed directly on the platform [85, 88]. Figures can be reproduced in a single click by importing the SED-ML file, which contains the simulation experiment to be run, the model that should be used in the simulation and plotting specifications [88]. Numerical integration for the simulation is performed using a Mathematica backend after the relevant experiment has been specified with a JavaScript Object Notation document [88]. Simulation experiments can be changed, analyzed and stored in the user workspace.

Model repositories

There are a number of model repositories now available to the biological modeling community. We will focus here on those repositories that are of specific interest to kinetic modeling. The reader is referred to the review by Dräger and Palsson [32] for additional information.

BioModels

With the development of SBML, it immediately became apparent that it would be possible to develop model repositories. The most well known of these is the BioModels database located [70] at the EBI near Cambridge, UK. BioModels is a service that aims to provide a central location where published models can be searched and downloaded. Of particular importance is that BioModels also employs a curation team so that users of the database can be sure that curated models work as described in the original publication. They also make sure that models are consistent with respect to MIRIAM guidelines [62]. Users can upload new models in SBML format, but the site offers users the ability to convert SBML into other formats such as MATLAB or XPP [38]. At the time of writing (2019), BioModels contains 653 curated models and 1023 uncurated models. Models include metabolic, signaling, gene and neurophysiological models. More recently, BioModels began storing whole-genome models. The NIH Center for Reproducible Biomedical Modeling has also started a service to curate models and is working closely with the BioModels team to harmonize curation standards. For dynamic kinetic models, BioModels is the primary resource, as well as additional models stored at JWS online.

BiGG

The BiGG database [65, 82] is a more recent repository that is geared towards storing structured genome-scale metabolic network reconstructions [99]. Whole genome-scale models are static biochemical network models that attempt to include every known reaction of an organism. Models are manually curated as with the BioModels database to ensure that the models are of high quality. An example of a heavily curated model is the yeast GEM, a model for Saccharomyces cerevisiae that has been under continuous development [94]. Models can be downloaded in standard SBML and used with constraint-based modeling tools such as COBRA [50, 59] or COBRApy [11, 36].

SEED

The SEED repository [31, 51] is managed by the US Department of Energy (DOE) as a database of genome-scale models relevant to the DOE mission. SEED is also part of the DOE Systems Biology Knowledgebase initiative [5]. The scope of the database is therefore much smaller. Recently, support has also come from the National Science Foundation to support the inclusion of plant-based genome-scale models [100]. As with the other repositories, SEED allows users to download their models in standard SBML. Note that SEED and BiGG only store constraint-based static models. For dynamic models of metabolism, users should use BioModels.

Although both BiGG and SEED only offer static models, such models are a necessary starting point for building dynamic models.

SABIO-RK

SABIO-RK [110] is a web-based database of kinetic rate laws. The kinetic laws held on the database have been extracted from the literature, which is a time-consuming manual process. The database is reaction orientated, and for each reaction entry, the database stores the rate equation, parameter values and environmental conditions under which the kinetics were measured. The database can also be accessed by a programmatic REpresentational State Transfer interface.

Data repositories

There are a range of more generic repositories that are not specific to systems biology but will cater to a wide variety of data types and disciplines although some are more widely used by specific communities than others.

SEEK

SEEK provides an integrated platform that supports storage of large and small data sets and mathematical model descriptions and preserves links between data and models that can explicitly identify and separate data used for model construction or model validation using an 'Investigation, Study, Assay' hierarchy [112, 113]. The SEEK platform also provides infrastructure to facilitate annotation of data and models using the appropriate minimum information standards and domain-specific ontological terms. Most data deposited on a SEEK platform is stored in Excel spreadsheets, which may already be used for data storage, and metadata can be extracted from these spreadsheets and stored in RDF to support searching data sets for key terms that describe the data. Additionally, the SEEK platform has templates for data and model uploads that provide functions to select the appropriate ontology term to describe the data or minimum information model, which are built on RightField [114], to embed lists of relevant terms into the spreadsheet for selection. The JWS simulator [88] is associated with the SEEK suite of tools, which enables online simulation of models, also supports annotation following the MIRIAM guidelines through the semantic SBML web services [69]. Additionally, annotated models can be readily viewed in the Systems Biology Graphical Notation [71] on JWS online [85].

Figshare

Figshare provides generic storage support for linking data sets to figures, graphics and pre-prints associated with research projects. Resources uploaded to Figshare may later be published in journal articles or might never be included in manuscripts submitted for publication [102]. Uploading all figures, especially those that do not support the narrative of published manuscripts ensures transparency and can prevent unnecessary investigations into dead-end research. Figshare provides a persistent identifier that supports long-term access to the uploaded resource [102]. Author names, categories and tags will be associated with the uploaded content and will allow other users to search for resource using relevant terms [102]. Academic uptake of Figshare for sharing and viewing data suggests that it is an effective generic platform for storing research content [105]. Recently, Figshare partnered with the NIH (see https://nih.figshare.com/) to provide data storage to NIH-partnered scientists working on research that is outside the scope of traditional repositories [60].

STRIDES

STRIDES (https://datascience.nih.gov/strides) is a new NIH initiative that allows NIH grant holders to access cloud space to store data and other artifacts. The initiative is very new and some questions remain, for example, what happens to any stored data once grant funding stops? STRIDES aims to follow the FAIR doctrine.

Zenodo

Zenodo is supported by CERN and was originally used by the high-energy physics community but has since become much

more broadly used by the scientific community. Researchers can deposit all manner of information, including data sets, software, report, etc. The site is well designed and is easy to search. In recent years, Zenodo has become a popular location for depositing documents and software related to the systems biology modeling community. One of its advantages is that all deposits are assigned a digital object identifier (DOI) that allows researchers to cite their work.

Dryad

Dryad is similar to Zenodo and, in fact, since 2019 has a partnership with Zenodo to work together. As with Zenodo, Dryad can store a wide variety of informational types and can issue DOIs for all deposits. Both Dryad and Zenodo offer a level of curation to prevent misuse of the repositories.

In summary, resources such as Dryad, Zenodo or FigShare offer researchers persistent locations on the internet where data and other information can be stored and linked to a published paper. There is no reason today that researchers need to store any research information on personal web sites that tend to have limited life spans. In this section, we have not mentioned GitHub and related sites as a possible place to keep research data. Generally speaking, sites such as GitHub are source code repositories and although they are used to store other data, it is not recommended. In addition, persistence is not guaranteed. If user accounts are closed, all the data is lost. We therefore do not recommend using sites such as GitHub to store models or data.

Conclusion

Providing the kinds of standards, resources and software described in the previous sections can go a long way to address the problems we face in repeatability and reproducibility of kinetic modeling experiments. A second and more difficult issue to address is a cultural one. As mentioned at the beginning of this article, there is no incentive to publish repeatable or reproducible models. Given the considerable pressures that are placed on researchers today, some aspects of our professionalism will inevitably be constrained by other priorities. Neither grant-awarding bodies, universities nor other research institutes give any credit to publishing reproducible science. A recent report on reproducibility by the National Academy of Sciences [79] refers to these pressures as 'misaligned incentives'. To quote from the report: 'Academic incentives—such as tenure, grant money and status—may influence scientists to compromise on good research practices. Faculty hiring, promotion and tenure decisions are often based in large part on the "productivity" of a researcher, such as the number of publications, number of citations and amount of grant money received' [37]. Thus, there is enormous institutional inertia to overcome. One approach that is being adopted by the Center for Reproducible Biomedical Modeling is to form partnerships with journals. The journals are the gatekeepers for much of our research, employing external reviews and providing editorial oversight. By partnering with journals, the the Center for Reproducible Biomedical Modeling acts as a 'fourth' reviewer of submitted manuscripts. The role of the center is to judge the quality of the computational experiments from the point of view of repeatability or reproducibility. Editors will receive a report and act upon it accordingly. Interestingly, the American Journal of Political Science (AJPS) has taken the lead by using a similar approach over the past 5 years. They state on their web site that: 'The corresponding author of a manuscript that is accepted for publication in the American Journal of Political Science must provide materials that are sufficient to enable interested researchers to verify all of the analytic results that are reported in the text and supporting materials'. When AJPS receives a manuscript to review, if the manuscript is accepted for publication, it is passed to a third party for what they term 'verification'. The manuscript is not released to the public until verification is satisfied. The journal provides very detailed instructions on how to prepare data and analyses that will pass the verification stage. For any journal, one might predict that imposing additional restrictions on publication would result in fewer manuscripts being received, which could be disadvantageous for the journal. In the case of AJPS, this appears not to be the case. If anything, the credibility of the journal has risen as a result, with the 2-year impact factor rising by a factor of two over the past 5 years [3]. Biostatistics is another journal that has taken the lead in publishing reproducible results. This journal places kite marks on the top-right corner of the first page of the published paper. Three levels can be indicated: D—if the data are freely available, C—if the authors' code is freely available and R—if both data and code are available. A reproducibility associate editor reviews papers and assigns the appropriate badge. A more detailed look at the role of badges can be found at [84]. With more journals and organizations such as the Center for Reproducible Biomedical Modeling getting involved with ensuring that published work is repeatable or reproducible, the hope is that these gentle nudges will slowly alter the publishing landscape. This is especially important for the large models that are now being generated by the kinetic modeling community.

In the absence of pressure by journals to induce researchers to provide reproducible research, individual researchers may improve their fields by insisting on reproducibility from the journal articles and faculty candidates they review and from their own work. Such a 'bottom-up' approach (advocated by Carey *et al.* [22]) not only can improve the science directly reviewed but also can provide more examples of higher-quality scientific results, which itself can provide pressure on journals and academic institutions to begin to change their policies and embrace new metrics of reproducibility.

Key Points

- The reproducibility of dynamic kinetic models is currently poor but is key to the further developments of the field.
- The current reward system within our institutions does not consider reproducibility an important output of a research project.
- A variety of standard formats, best practices and tooling is available to help researchers publish reproducible work.
- Working with journals to help authors publish reproducible work may be the way forward to improve the publication of reproducible research.

Acknowledgements

We acknowledge the many fruitful discussions we have had with the COMBINE community (http://co.mbine.org/) that

has been largely responsible for many of the developments described in this article.

Funding

The research reported in this publication was supported by National Institute of General Medical Sciences and National Institute of Biomedical Imaging and Bioengineering of the National Institutes of Health under award number: R01GM123032 and P41GM109824, respectively, as well as National Science Foundation Civil, Mechanical and Manufacturing Innovation award 1933453. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH, National Science Foundation, the University of Washington or the Center for Reproducible Biomedical Modeling.

References

1. Adams RR. SED-ED, a workflow editor for computational biology experiments written in SED-ML. *Bioinformatics* 2012;**28**(8):1180–1.
2. Alves R, Antunes F, Salvador A. Tools for kinetic modeling of biochemical networks. *Nat Biotechnol* 2006;**24**(6):667–73.
3. Notes from the editors. American Political Science Review. 2019;**113**(4):iii–vii.
4. Antezana E, Blondé W, Egaña M, *et al*. BioGateway: a semantic systems biology tool for the life sciences. *BMC Bioinform* 2009;**10**(Suppl 10):S11.
5. Arkin A, Cottingham R, Henry C, *et al*. KBase: The United States Department of Energy Systems Biology Knowledgebase. *Nat Biotechnol* 2018;**36**:566–9. https://doi.org/10.1038/nbt.4163
6. Association for Computing Machinery. Artifact review and badging. 2018.
7. Ashley GS. The war over supercooled water. *Phys Today* 2018. doi: 10.1063/PT.6.1.20180822a.
8. Baichoo S, Souilmi Y, Panji S, *et al*. Developing reproducible bioinformatics analysis workflows for heterogeneous computing environments to support African genomics. *BMC Bioinform* 2018;**19**(1):457.
9. Baker M. 1,500 scientists lift the lid on reproducibility. *Nature* 2016;**533**(7604):452–4.
10. Barba LA. Terminologies for reproducible research. *ArXiv*, abs/1802.03311. 2018.
11. Becker SA, Feist AM, Mo ML, *et al*. Quantitative prediction of cellular metabolism with constraint-based models: the COBRA toolbox. *Nat Protoc* 2007;**2**(3):727–38.
12. Bergmann F. SED-ML script language. *Nat Preced* 2011;1.
13. Bergmann FT, Adams R, Moodie S, *et al*. COMBINE archive and OMEX format: one file to share all information to reproduce a modeling project. *BMC Bioinform* 2014;**15**(1):369.
14. Bergmann FT, Cooper J, König M, *et al*. Simulation experiment description markup language (SED-ML) level 1 version 3 (L1V3). *J Integr Bioinform* 2018;**15**(1):20170086.
15. Bergmann FT, Keating SM, Gauges R, *et al*. SBML level 3 package: render, version 1, release 1. *J Integr Bioinform* 2018;**15**(1). doi: 10.1515/jib-2017-0078.
16. Neupane JB, Neupane RP, Luo Y, *et al*. Characterization of leptazolines A-D, polar oxazolines from the cyanobacterium Leptolyngbya sp., reveals a glitch with the 'Willoughby–Hoye' scripts for calculating NMR chemical shifts. *Org Lett* 2019;**21**(20):8449–53.
17. Blainey P, Krzywinski M, Altman NS. Points of significance: replication. *Nat Methods* 2014;**11**(9):879–80.
18. Bornstein BJ, Keating SM, Jouraku A, *et al*. LibSBML: an API library for SBML. *Bioinformatics* 2008;**24**(6):880–1.
19. Bouteiller J-MC, Feng Z, Onopa A, *et al*. Maximizing predictability of a bottom-up complex multi-scale model through systematic validation and multi-objective multi-level optimization. In: *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER), pp. 300–303*. IEEE, 2015.
20. Cannistra C, Medley K, Sauro H. Simplesbml: a python package for creating and editing SBML models. *BioRxiv* 2015;030312. doi: 10.1101/030312.
21. Yang C, Gillespie DT, Petzold LR. Efficient step size selection for the tau-leaping simulation method. *J Chemi Phys* 2006;**124**(4):044109. doi: 10.1063/1.2159468.
22. Carey MA, Dräger A, Papin JA, *et al*. Community standards to facilitate development and address challenges in metabolic modeling. *BioRxiv* 2019;700112. doi: 10.1101/700112.
23. Chamberlain R, Schommer J. Using docker to support reproducible research. *DOI: https://doi. org/10.6084/m9. figshare*, 1101910:44, 2014.
24. Chen TY, Ho JWK, Liu H, *et al*. An innovative approach for testing bioinformatics programs using metamorphic testing. *BMC Bioinform* 2009;**10**(1):24.
25. Choi K, Medley JK, König M, *et al*. Tellurium: an extensible Python-based modeling environment for systems and synthetic biology. *Biosystems* 2018;**171**:74–9.
26. Choi K, Smith LP, Medley JK, *et al*. PhraSED-ML: a paraphrased, human-readable adaptation of SED-ML. *J Bioinform Comput Biol* 2016;**14**(06):1650035.
27. Choi K, Smith LP, Medley JK, *et al*. Phrased-ml: a paraphrased, human-readable adaptation of sed-ml. *J Bioinform Comput Biol* 2016;**14**(06):1650035.
28. Copeland WB, Bartley BA, Chandran D, *et al*. Computational tools for metabolic engineering. *Metab Eng* 2012;**14**(3):270–80.
29. Cowan AE, Mendes P, Blinov ML. ModelBricks—modules for reproducible modeling improving model annotation and provenance. *NPJ Syst Biol Appl* 2019;**5**(1):37.
30. Deelman E, Vahi K, Juve G, *et al*. Pegasus, a workflow management system for science automation. *Future Gener Comput Syst* 2015;**46**:17–35.
31. DeJongh M, Formsma K, Boillot P, *et al*. Toward the automated generation of genome-scale metabolic networks in the SEED. *BMC Bioinform* 2007;**8**(1):139.
32. Dräger A, Palsson BO. Improving collaboration by standardization efforts in systems biology. *Front Bioeng Biotechnol* 2014;**2**:1–20.
33. Dräger A, Rodriguez N, Dumousseau M, *et al*. JSBML: a flexible java library for working with SBML. *Bioinformatics* 2011;**27**(15):2167–8.
34. Easterbrook SM. Open code for open science? *Nat Geosci* 2014;**7**(11):779.
35. Ebrahim A, Almaas E, Bauer E, *et al*. Do genome-scale models need exact solvers or clearer standards? *Mol Syst Biol* 2015;**11**(10):831.
36. Ebrahim A, Lerman JA, Palsson BO, *et al*. COBRApy: cOnstraints-based reconstruction and analysis for python. *BMC Syst Biol* 2013;**7**:74.
37. Edwards MA, Roy S. Academic research in the 21st century: maintaining scientific integrity in a climate of perverse incentives and Hypercompetition. 2017;51–61.
38. Ermentrout BX. *Computational Systems Neurobiology*. Dordrecht: Springer Netherlands, 2012, 519–31.

39. Ewald R, Uhrmacher AM. SESSL: a domain-specific language for simulation experiments. *ACM Trans Modeling Comput Simul* 2014;**24**(2):1–25.
40. Fanelli D. Opinion: is science really facing a reproducibility crisis, and do we need it to? *Proc Natl Acad Sci* 2018;**115**(11):2628–31.
41. Gauges R, Rost U, Sahle S, *et al*. The systems biology markup language (SBML) level 3 package: layout, version 1 Core. *J Integr Bioinform* 2015;**12**(2):267.
42. Gennari JH, Neal ML, Galdzicki M, *et al*. Multiple ontologies in action: composite annotations for biosimulation models. *J Biomed Inform* 2011;**44**(1):146–54.
43. Gibson MA, Bruck J. Efficient exact stochastic simulation of chemical systems with many species and many channels. *J Phys Chem A* 2000;**104**(9):1876–89.
44. Gilbert J, Pearcy N, Norman R, *et al*. Gsmodutils: a python based framework for test-driven genome scale metabolic model development. *Bioinformatics* 2019;**35**(18): 3397–403.
45. Gillespie CS, Wilkinson DJ, Proctor CJ, *et al*. Tools for the SBML community. *Bioinformatics* 2006;**22**(5):628–9.
46. Gillespie DT. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *J Comput Phys* 1976;**22**(4):403–34.
47. Goble CA, Bhagat J, Aleksejevs S, *et al*. myExperiment: a repository and social network for the sharing of bioinformatics workflows. *Nucleic Acids Res* 2010;**38**(Web Server issue):W677–82.
48. Goecks J, Nekrutenko A, Taylor J, *et al*. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol* 2010;**11**(8):R86.
49. Stanley G, Herbert SM. Standards, platforms, and applications. In: Kriete A, Eils R (eds). *Computational Systems Biology: From Molecular Mechanisms to Disease, chapter 8*. Academic Press, 2013, 133–67.
50. Heirendt L, Arreckx S, Pfau T, *et al*. Creation and analysis of biochemical constraint-based models using the cobra toolbox v. 3.0. *Nat Protoc* 2019;**14**(3):639–702.
51. Henry CS, DeJongh M, Best AA, *et al*. High-throughput generation, optimization and analysis of genome-scale metabolic models. *Nat Biotechnol* 2010;**28**(9):977.
52. Hettne KM, Wolstencroft K, Belhajjame K, *et al*. Best practices for workflow design: how to prevent workflow decay. In: *SWAT4LS*, 2012.
53. Hindmarsh AC, Brown PN, Grant KE, *et al*. SUNDIALS: suite of nonlinear and differential/algebraic equation solvers. *ACM Trans Math Softw* 2005;**31**(3):363–96.
54. Hoksza D, Gawron P, Ostaszewski M, *et al*. Closing the gap between formats for storing layout information in systems biology. *Brief Bioinform* 2019;(4):1249–60.
55. Hoops S, Sahle S, Gauges R, *et al*. COPASI–a complex pathway simulator. *Bioinformatics* 2006;**22**(24):3067–74.
56. Hucka M, Finney A, Sauro HM, *et al*. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 2003;**19**(4):524–31.
57. Hucka M, Bergmann FT, Dräger A, *et al*. The systems biology markup language (SBML): language specification for level 3 version 2 core. *J Integr Bioinform* 2018;**15**(1): 20170081.
58. Hutton C, Wagener T, Freer J, *et al*. Most computational hydrology is not reproducible, so is it really science? *Water Resour Res* 2016;**52**(10):7548–55.
59. Hyduke D, Schellenberger J, Que R, *et al*. Cobra toolbox 2.0. *Protoc Exch* 2011;**22**(10.1038):1–44.
60. Hyndman A. *Figshare announces data repository partnership with the National Institutes of Health to store and reuse research data*. 2019. https://figshare.com/blog/Figshare_announces_data_repository_partnership_with_the_National_Institutes_of_Health_to_store_and_reuse_research_data/518.
61. Jia G, Stephanopoulos G, Gunawan R. Ensemble kinetic modeling of metabolic networks from dynamic metabolic profiles. *Metabolites* 2012;**2**(4):891–912.
62. Juty N, Le Novere N, Laibe C. Identifiers.org and MIRIAM registry: community resources to provide persistent identification. *Nucleic Acids Res* 2012;**40**(D1):D580–6.
63. Kacser H, Burns JA, Kacser H, *et al*. The control of flux. *Biochem Soc Trans* 1995;**23**(2):341–66.
64. Keller R, Dörr A, Tabira A, *et al*. The systems biology simulation core algorithm. *BMC Syst Biol* 2013;**7**(1):55.
65. Zachary A. King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A. Lerman, Ali Ebrahim, Bernhard O. Palsson, and Nathan E. Lewis. BiGG models: a platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Res* 2016;**44**(D1): D515–22.
66. Kohl M. *Standards, Databases, and Modeling Tools in Systems Biology*. Totowa, NJ: Humana Press, 2011, 413–427.
67. König M. Executable simulation model of the liver. *BioRxiv* 2020;2020.01.04.894873. https://doi.org/10.1101/2020.01.04.894873.
68. Köster J, Rahmann S. Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics* 2012;**28**(19):2520–2.
69. Krause F, Uhlendorf J, Lubitz T, *et al*. Annotation and merging of SBML models with semanticSBML. *Bioinformatics* 2009;**26**(3):421–2.
70. N. Le Novere. BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Res* 2006;**34**(90001):D689–91.
71. Nicolas Le Novere, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I Aladjem, Sarala M Wimalaratne, *et al*. The systems biology graphical notation. *Nat Biotechnol* 2009;**27**(8):735.
72. Liebermeister W, Krause F, Uhlendorf J, *et al*. Semanticsbml: a tool for annotating, checking, and merging of biochemical models in sbml format. *Nat Preced* 2009;1. https://doi.org/10.1038/npre.2009.3093.1.
73. Lieven C, Beber ME, Olivier BG, *et al*. Memote for standardized genome-scale metabolic model testing. *Nat Biotechnol* 2020;**38**(3):272–6.
74. Maggioli F, Mancini T, Tronci E. SBML2Modelica: integrating biochemical models within open-standard simulation ecosystems. *Bioinformatics* 2019;**36**(7):2165–72. https://doi.org/10.1093/bioinformatics/btz860.
75. J. Kyle Medley, Kiri Choi, Matthias König, Lucian Smith, Stanley Gu, Joseph Hellerstein, Stuart C. Sealfon, and Herbert M. Sauro. Tellurium notebooks—an environment for reproducible dynamical modeling in systems biology. *PLoS Comput Biol* 2018;**14**(6):e1006220.
76. Medley JK, Hellerstein J, Sauro HM. Libsbmljs—enabling web–based SBML tools. *Bio Systems* 2020;**195**:104150. doi: 10.1016/j.biosystems.2020.104150.
77. P Mendes. GEPASI: a software package for modelling the dynamics, steady states and control of biochemical and other systems. *Comput Appl Biosci* 1993;**9**(5):563–71.

78. Mitra ED, Suderman R, Colvin J, *et al*. Pybionetfit and the biological property specification language. *IScience* 2019;**19**:1012–36.
79. Engineering National Academies of Sciences and Medicine. *Reproducibility and Replicability in Science*. Washington, DC: The National Academies Press, 2019.
80. Neal ML, Thompson CT, Kim KG, *et al*. Semgen: a tool for semantics-based annotation and composition of biosimulation models. *Bioinformatics* 2019;**35**(9):1600–2.
81. Maxwell Lewis Neal, Matthias König, David Nickerson, Göksel Misirli, Reza Kalbasi, Andreas Dräger, Koray Atalag, Vijayalakshmi Chelliah, Michael T. Cooling, Daniel L. Cook, Sharon Crook, Miguel De Alba, Samuel H. Friedman, Alan Garny, John H. Gennari, Padraig Gleeson, Martin Golebiewski, Michael Hucka, Nick Juty, Chris Myers, Brett G. Olivier, Herbert M. Sauro, Martin Scharm, Jacky L. Snoep, Vasundra Touré, Anil Wipat, Olaf Wolkenhauer, and Dagmar Waltemath. Harmonizing semantic annotations for computational models in biology. *Brief Bioinform* 2019;**20**(2):540–50.
82. Norsigian CJ, Pusarla N, McConn JL, *et al*. Bigg models 2020: multi-strain genome-scale models and expansion across the phylogenetic tree. *Nucleic Acids Res* 2020;**48**(D1):D402–6.
83. Nowak U, Weimann L. A family of Newton codes for systems of highly nonlinear equations. *Technical Report*.
84. Nüst D, Lohoff L, Einfeldt L, *et al*. Guerrilla badges for reproducible geospatial data science (AGILE 2019 short paper). *AGILE 2019* 2019.
85. B. G. Olivier and J. L. Snoep. Web-based kinetic modelling using JWS online. *Bioinformatics* 2004;**20**(13):2143–4.
86. Brett G. Olivier and Frank T. Bergmann. SBML level 3 package: flux balance Constraints version 2. *J Integr Bioinform* 2018;**15**(1):20170082.
87. Paffenroth R, Doedel EJ. The auto2000 command line user interface. In: *Proceedings of the 9th International Python Conference, pp. 233–241*, 2001.
88. Martin Peters, Johann J. Eicher, David D. van Niekerk, Dagmar Waltemath, and Jacky L. Snoep. The JWS online simulation database. *Bioinformatics* 2017;btw831.
89. Linda Petzold. Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations. *SIAM J Sci Stat Comput* 1983;**4**(1):136–48.
90. Plesser HE. Reproducibility vs. replicability: a brief history of a confused terminology. *Front Neuroinform* 2018; **11**:76.
91. Porubsky VL. Tellurium & libRoadRunner tutorial for the COMBINE & de.NBI 2019 workshop on modeling and simulation tools in systems biology. *Technical Report*. University of Washington, 2019.
92. De Roure D, Manuel J, Hettne K, *et al*. Towards the preservation of scientific workflows. In: *Procs. of the 8th International Conference on Preservation of Digital Objects (iPRES 2011)*. ACM, 2011.
93. Salvadores M, Alexander PR, Musen MA, *et al*. BioPortal as a dataset of linked biomedical ontologies and terminologies in RDF. *Semant Web* 2013;**4**(3):277–84.
94. Sánchez BJ, Nielsen J. Genome scale models of yeast: towards standardized evaluation and consistent omic integration. *Integr Biol* 2015;**7**(8):846–58.
95. Sauro HM. *33 JARNAC: a system for interactive metabolic analysis*. Technical Report, 2000.
96. Sauro HM, Bergmannm F. Software tools for systems biology. In: Liu ET, Lauffenburger DA (eds). *Systems Biomedicine: Concepts and Perspectives, chapter 12*. Academic Press, 2009, 289–314.
97. Sauro HM, Fell DA. SCAMP: a metabolic simulator and control analysis program. *Math Comput Model* 1991;**15**(12): 15–28.
98. Sauro HM. *Systems Biology: An Introduction to Metabolic Control Analysis*. Seattle. USA: Ambrosius Publishing, 2018.
99. Schellenberger J, Park JO, Conrad TM, *et al*. BiGG: a biochemical genetic and genomic knowledgebase of large scale metabolic reconstructions. *BMC Bioinform* 2010;**11**(1): 213.
100. Samuel M.D. Seaver, Svetlana Gerdes, Océane Frelin, Claudia Lerma-Ortiz, Louis M.T. Bradbury, Rémi Zallot, Ghulam Hasnain, Thomas D. Niehaus, Basma El Yacoubi, Shiran Pasternak, Robert Olson, Gordon Pusch, Ross Overbeek, Rick Stevens, Valérie De Crécy-Lagard, Doreen Ware, Andrew D. Hanson, and Christopher S. Henry. High-throughput comparison, functional annotation, and metabolic modeling of plant genomes using the PlantSEED resource. *Proc Natl Acad Sci U S A* 2014;**111**(26):9645–50.
101. Segura S, Hierons RM, Benavides D, *et al*. Automated test data generation on the analyses of feature models: a metamorphic testing approach. In: *2010 Third International Conference on Software Testing, Verification and Validation, pp. 35–44*. IEEE, 2010.
102. J FigShare Singh. *J Pharmacol Pharmacother* 2011;**2**(2):138–9.
103. L. P. Smith, F. T. Bergmann, D. Chandran, and H. M. Sauro. Antimony: a modular model definition language. *Bioinformatics* 2009;**25**(18):2452–4.
104. Endre T. Somogyi, Jean Marie Bouteiller, James A. Glazier, Matthias König, J. Kyle Medley, MacIej H. Swat, and Herbert M. Sauro. LibRoadRunner: a high performance SBML simulation and analysis library. *Bioinformatics* 2015;**31**(20):3315–21.
105. Thelwall M, Kousha K. Figshare: a universal repository for academic resource sharing?. *Online Inf Rev* 2016;**40**(3):333–46. doi: 10.1108/OIR-06-2015-0190.
106. Andra Waagmeester, Martina Kutmon, Anders Riutta, Ryan Miller, Egon L. Willighagen, Chris T. Evelo, and Alexander R. Pico. Using the semantic web for rapid integration of WikiPathways with other biological online data resources. *PLoS Comput Biol* 2016;**12**(6):e1004989.
107. Waltemath D, Adams R, Bergmann FT, *et al*. Reproducible computational biology experiments with SED-ML—the simulation experiment description markup language. *BMC Syst Biol* 2011;**5**(1):198.
108. Liqing Wang, Inanç Birol, and Vassily Hatzimanikatis. Metabolic control analysis under uncertainty: framework development and case studies. *Biophys J* 2004;**87**(6): 3750–63.
109. Wilkinson MD, Dumontier M, Aalbersberg IJJ, *et al*. The fair guiding principles for scientific data management and stewardship. *Sci Data* 2016;**3**:160018.
110. Wittig U, Kania R, Bittkowski M, *et al*. Data extraction for the reaction kinetics database SABIO-RK. *Perspect Sci* 2014; **1**(1–6):33–40.
111. Wolstencroft K, Haines R, Fellows D, *et al*. The Taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic Acids Res* 2013;**41**(Web Server issue):W557–61.
112. Katherine Wolstencroft, Stuart Owen, Olga Krebs, Quyen Nguyen, Natalie J Stanford, Martin Golebiewski, Andreas Weidemann, Meik Bittkowski, Lihua An, David Shockley,

Jacky L. Snoep, Wolfgang Mueller, and Carole Goble. SEEK: a systems biology data and model management platform. *BMC Syst Biol* 2015;**9**(1):33.

113. Wolstencroft K, Owen S, Du Preez F, *et al*. The SEEK: a platform for sharing data and models in systems biology. *Methods in Enzymology, Vol*. 500. SAN DIEGO, USA: Academic Press Inc., 2011, 629–655.
114. Wolstencroft K, Owen S, Horridge M, Krebs O, Mueller W, Jacky L. Snoep, Franco du Preez, and Carole Goble. RightField: embedding ontology annotation in spreadsheets. *Bioinformatics* 2011;**27**(14):2021–2.
115. Wright S. Physiological and evolutionary theories of dominance. *Am Nat* 1934;**68**(714):24–53.