Reproducibility in Computational Neuroscience Models and Simulations

Robert A. McDougal, Anna S. Bulanova, William W. Lytton

***Abstract*—Objective: Like all scientific research, computational neuroscience research must be reproducible. Big data science, including simulation research, cannot depend exclusively on journal articles as the method to provide the sharing and transparency required for reproducibility.**

Methods: Ensuring model reproducibility requires the use of multiple standard software practices and tools, including version control, strong commenting and documentation, and code modularity.

Results: Building on these standard practices, model sharing sites and tools have been developed that fit into several categories: 1. standardized neural simulators, 2. shared computational resources, 3. declarative model descriptors, ontologies and standardized annotations; 4. model sharing repositories and sharing standards.

Conclusion: A number of complementary innovations have been proposed to enhance sharing, transparency and reproducibility. The individual user can be encouraged to make use of version control, commenting, documentation and modularity in development of models. The community can help by requiring model sharing as a condition of publication and funding.

Significance: Model management will become increasingly important as multiscale models become larger, more detailed and correspondingly more difficult to manage by any single investigator or single laboratory. Additional *big data* management complexity will come as the models become more useful in interpreting experiments, thus increasing the need to ensure clear alignment between modeling data, both parameters and results, and experiment.

***Index Terms*—Reproducibility, computational neuroscience, model sharing, simulator, annotation.**

I. Introduction

THE complexity of the brain provides a key challenge for neuroscience research, for computational modeling, and for data science. Experimentalists have probed the brain at many scales, from electron microscopy studies of synapses [1], through axons [2], whole neurons and brain slices, up to an awake behaving animal [3]. Many experiments reach from the lowest to highest scales, for example evaluating performance on a memory task as a drug modifies ion channel activity at the molecular level.

Computational neuroscience is a specialized branch of computational systems biology. It provides explicit multiscale models that can unify experimental observations and build novel theoretical frameworks. A century ago, work by Lapicque led to the development of integrate-and-fire models [4]. A half century later, Hodgkin and Huxley provided a detailed multiscale biophysical model of the squid axon [2], paving the way for subsequent advances in computational neuroscience. Early on, simulations were generally one-off or highly specialized and restricted to one laboratory, leading to difficulties with reproducibility. In the past decades, several major general-purpose neuroscience simulators have been developed, providing some degree of standardization and enhancing reproducibility.

R.A. McDougal is with the Department of Neuroscience, Yale University, New Haven, CT, 06520 USA e-mail: robert.mcdougal@yale.edu.

A.S. Bulanova is with SUNY Downstate and Yale University. W.W. Lytton is with SUNY Downstate and Kings County Hospital.

Manuscript received November 3, 2015; revised February 1, 2016; accepted March 4, 2016.

A. Overview

We start by describing general issues of reproducibility (providing close-enough results) and replicability (providing precisely the same results – a replica) as they specifically relate to the computational neuroscience enterprise. We then describe a number of specific tools, techniques and concepts that have been developed to encourage sharing, transparency, reproducibility, and replication. The development of a number of *standard neural simulators* means that groups of researchers now speak a common language, reducing babel. *Shared computational resources* such as supercomputers now provide access to previously restricted computational resources. *Declarative model descriptions* have been developed that separate a model from the specific implementations, again providing a common language that now extends across simulators. *Repositories and sharing standards* for code, data, and models provide assurance that information is shared appropriately, and as completely as possible. We consider each of these topics in turn, focusing on specific examples and comparing related tools.

B. Reproducibility and Replicability

Reproducibility, or rather its lack, is a major problem in both experimentation and modeling. Results that cannot be reproduced cannot be used for subsequent research to further scientific progress. Difficulty in reproducing a model (or experiment) does not imply misconduct or poor quality research – typos happen, and key details are often left out of a paper. Data-sharing difficulties are common to all of *big data science* (distinguishable from *big science* where the focus is on the large groups of coworkers or collaborators on a project). Simulation suffers from big-data problems on both ends, both in the data that goes into the project and in the data that comes out. Insufficient detail at the input side – the model definition – is a particular problem for simulation. If one looks at a

modeling paper, full simulation code is almost never included, key equations may be omitted or assumed (*e.g.,* abutment between electrical compartments can be handled in many different ways), and code cannot always be precisely replicated even when all equations are given (*e.g.,* different randomizers). Lack of reproducibility impairs and delays scientific progress, making it difficult or impossible for subsequent researchers to stand on the proverbial shoulders.

What does it mean for a neuroscience model to be reproducible? The words *reproducible*, *repeatable* and *replicable* are used in scientific discussions in multiple different ways, sometimes with contradictory definitions. Drummond [5] and Crook et al. [6] draw a clear distinction between reproducible and replicable. We follow their terminology and define:

- A *replicable simulation* can be repeated exactly (*e.g.,* by rerunning the source code on the same computer).
- A *reproducible simulation* can be independently reconstructed based on a description of the model.

It follows that a replication should give precisely identical results, while a reproduction will give results which are similar but often not identical.

In general, a reproducible simulation offers more insight into model design and meaning than does a model that is simply replicable. At one extreme, an executable (machine code) version of a simulation is entirely replicable on the architecture of origin, but offers no insight into the design of the model, and no practical possibility of extending it. Source code is the best means for providing a model that is replicable across architectures. Source code can also provide a moderate level of insight into a model, depending in large part on how well structured, well commented, and well documented it is. Some models are by necessity abstrusely coded, particularly large models that are designed to run speedily on specific architectures. Furthermore, software only provides an approximation for an underlying abstract mathematical description (called the *ansatz* in physics) that is approximated by software using numerical integration or other numerical approximation techniques. At the next level of abstraction, the ansatz is based on an integrated conceptual model. The conceptual model is formed out of multiple hypotheses, explicit and implicit, and a necessarily incomplete set of experimental observations, after deciding what needs to be included and what can be left out. (Experiments are themselves constrained by governing hypotheses that determine what is looked for, what is measured, and what is observed.) This group of selected hypotheses and observations underlies the conceptual model – it is our view of how the biology works. Along this spectrum, from experiment to machine code, one sees a monotonic increase in replicability, while reproducibility is governed by an inverted-U (Fig. 1). Peak reproducibility will generally lie at the level of the mathematical model or algorithm.

Reproducibility and replicability are external measures that reflect a user's view of similarities in dynamics between an original reference and a reproduction. By contrast, *robustness* is generally taken to be an internal measure between different models, with slight changes in parameters, that assesses the ability to preserve similar dynamics despite these changes. A model that is not robust will not be readily reproducible, since implementation differences are likely to introduce small changes that will have effects similar to small changes in parameters. If the simulation is intrinsically robust then reproducibility, reproducing the result of a simulation with a different implementation, provides additional evidence of robustness, [5], [9], [10] and demonstrates that a result is not an artifact of implementation details or, worse yet, of implementation bugs. An alternative implementation may have improved performance characteristics that may make it useful to merge with the original implementation to build an improved composite implementation [10].

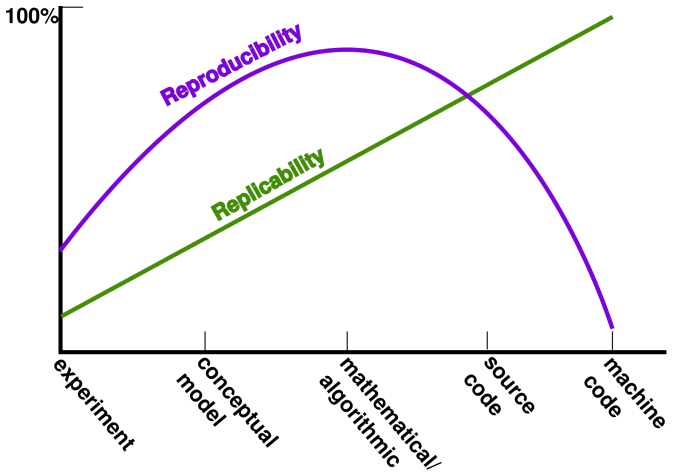


Fig. 1. Reproducibility ascends as one moves from experimental data to a peak coinciding with peak abstraction at the mathematical level, then declines again as one proceeds to instantiate that model. Meanwhile, replicability, defined here as the ability to produce precisely the same result, increased gradually. Replicability is nominally impossible at the level of experiment: experiments that measure analog values won't yield the same precise value twice. (Too many experiments also turn out to be irreproducible [7].) Whether a conceptual model is replicated or reproduced is hard to say – the concept exists in the users mind, only represented in schematic on the back of an envelope or other location. Similarly, although mathematical models and algorithms offer both excellent reproducibility and replicability, they may sometimes be inaccessible, in rare cases overwhelming even the best-prepared minds [8].

Despite every effort to control conditions, full replicability is an impossible goal in experimental research [5]. Human or animal behavioral research is the least replicable since it is never possible to control what the subject is thinking [11]. Computational research, by contrast, offers the potential for real replicability due to the deterministic nature of computers. Even here, there remains risk of non-replicability due to the small but nonzero risk of bit-flipping soft errors [12]. Such small errors can have significant effects in large simulations run for longer time periods, and in simulations run across multiple processors. Neuronal network simulations are particularly vulnerable to showing major alterations due to a bit flip or roundoff error, since a small shift in threshold crossing-time will percolate through the network and produce vast alterations in spike timing after one or more seconds of virtual time. Although computational replicability is seen by some as a potentially wasteful distraction from the broader scientific goal of reproducibility [5], it is important that models be replicable since they are often re-used as the building blocks for larger

models.

C. Challenges to replicability and reproducibility

The deterministic nature of digital computers and the precision of mathematical descriptions does permit us to avoid the kinds of difficulties faced by experimentalists, such as failure to reproduce experiments due to use of animals of different gender, or different age, or different strains. Still, there are many ways in which modeling can have problems. In particular, the complexity of detailed simulations introduces new challenges of adequate documentation, training and instructions to allow a program to be run correctly. Additionally, complex simulations are typically built within the context of complex simulation packages which are themselves difficult to master.

Computational simulations are built on explicitly stated mathematical rules – typically differential equations, but sometimes difference equations, or state-machine descriptions. As in other fields, research is primarily communicated via journal articles, which require the model to be expressed in a combination of equations and verbal descriptions. Although some journals permit extensive appendices, page limits generally prevent a complex model from being fully described. Author errors may be augmented by type-setting errors – *e.g.,* subscripts or superscripts that end up at line level. Additionally, the description of the model used in one figure may not be the same as that used for another.

Another difficulty occurs when models require specialized hardware or software, generally to optimize for speed. Large-scale network simulations may require High Performance Computers (HPCs), or Graphical Processing Units (GPUs) [13], [14]. Still more problematic are simulations that use highly specialized systems such as neuromorphic chips [15], [16], or SpiNNaker chips [17], [18], hardwares that are not commercially available. Similarly, at the software level, simulations may be built atop commercial software such as Matlab, posing a cost barrier. Older simulators, once used for neuroscience research may no longer be available, or may be available but not runnable under modern operating systems, or may have never been generally distributed – *e.g.,* ASTAP [19], [20]. Even when software is free, open-source, current, and readily downloadable, it may be inaccessible if the installation is complicated, or if it will only compile or run on particular platforms.

In order to determine if a model has been successfully replicated or reproduced, one wants to compare with the output of the original simulations shown in a figure. These data are not typically made available with the simulation, so that one is restricted to eyeballing the similarity with the figure. When the full data are available, there are several reasons why results may not agree precisely [6]. Quantitative differences can arise due to 32- vs 64-bit calculations, or differences in numerical methods. As noted above, the highly nonlinear nature of neural dynamics with discontinuities at thresholds means that a minor variation at one time step can produce large quantitative differences over time, especially in network models.

Deterministic results may also differ in the case of parallel simulations, if variable communication latencies and system overhead lead to processors completing tasks in a different order relative to each other in subsequent runs. This type of parallel error should be avoidable through the use of locks, but these are not always placed appropriately. Simulations on unusual parallel machines, *e.g.,* the SpiNNaker architecture, are globally asynchronous [18] and can produce variations between runs as spikes arrive from presynaptic cells in potentially different orders.

If a model is built in part by random number generation, care is necessary to ensure that the random sample may be reproduced on varying numbers of processors. In these models, the random seeds are part of the description, and must be recorded and shared to allow replicability.

II. Neuroscience simulators

The widespread use of neuroscience simulators improves model reproducibility by allowing models to be coded at an abstract level that maps more directly to the biology [21]. This abstraction separates the model from the numerics, making the elements of the model more identifiable and understandable. Because simulator programming languages are structured to be understandable, a model written carefully in such a simulator can approach the organizational clarity of a *Declarative model description* (Section IV). Common neuroscience concepts, like an unbranched section of dendrite or a chemical reaction, will be expressed in a consistent way across simulations [22], allowing the biology to be understood by reading the code [23]. This contrasts with a simulation written in a general-purpose language such as C, Fortran, Java or Matlab, which may not have any easily discerned relation to familiar neurobiological entities.

Utilizing a neuroscience simulator also reduces many problems introduced by implementing (re-implementing) from scratch [24]. It allows the basic software engineering to be delegated to software specialists. Most current neuroscience simulators are open source. This allows independent development by users, and permits investigation of underlying causes of any problems that arise. The algorithms, numerics and others, are generally published, peer-reviewed and extensively tested by users, providing increased confidence in their validity. Most of these simulators also allow an intermediate approach to be used when developing novel models that do not clearly fit into the existing categories provided by that simulator; one can integrate custom code written in a general-purpose language into a simulation described in the overlaying simulator [25].

Simulators invariably grow in complexity over years of use, particularly as growing numbers of users request additional features. This growth will affect the user interface as well as altering simulator scope and the numerics backend. Typically, a simulator will start by providing the user with the ability to define objects such as channels and cells, and to set model parameters, in an independent parameter file format. This forms a very basic declarative language which then grows. The language gradually takes on more of the attributes of a general computer language, including flow control and data structures.

For a long time, each neuroscience simulator had its own language. In recent years, neuroscience simulator designers have converged on Python as a *lingua franca* [26], [27]. This convergence has greatly improved cross-simulator readability and has also enabled linking of simulations across two or more simulators (Section II-E). Most major simulators now either use Python exclusively or as an alternative to an older language [28]–[36].

In addition to a parameter or scripting language, simulators often provide a graphical user interface (GUI) to help define or visualize models. Models constructed by GUI tools are usually viewable by similar tools, providing a consistent, predictable means of entering and extracting information. This consistency facilitates understanding and reproducibility. The SNNAP simulator [37], [38] is unusual in that it provides all model specification via its GUI. VCell [39] and NEURON [40] are more typical in that they allow many aspects of simulations to be constructed by GUI tools, but also allow more complicated simulations or added features to be entered by programmatic specification. PSICS [41] takes a slightly different approach by separating its GUI tool ICING from the simulator, allowing it to be used in other contexts.

Simulators may also provide ways of extracting information graphically even though a model was entered via code. GENESIS [42] provides a *showmsg* command that reports what parts of a simulation are affected by a given object. NEURON's ModelView [43] extracts the structure of a model, however created, from NEURON's internal data structures and displays it graphically. Similar functionality is available for some models on the ModelDB repository (modeldb.yale.edu) [23], [44].

The inherent multiscale nature of computational neuroscience was already identified in the founding document that provided the name [45]. One thing that makes computational biology different from other simulation areas such as physics and chemistry is the need for a multiscale approach, due to the impossibility of fully abstracting a lower level in the one above. To a greater or lesser extent, all of the simulators described here provide multiscale capabilities, generally at least providing the ability to simulate at the subcellular and cellular scales, or at the cellular and network scales. We start by describing the simulators that provide the broadest scale-coverage, and then describe some of the simulators that provide more detailed coverage by specializing at particular scales. Finally we mention tools that provide multiscale coverage by linking simulators.

A. Multiscale simulators

GENESIS [42], MOOSE [28], [46], and NEURON [40] are simulators that aspire to the largest multiscale range, from molecular [22], [47], [48] to large networks [49], [50], with the ability to simulate across these multiple scales [51]. These simulators generally encourage the use of multicompartmental models of neurons, utilizing neuroanatomically-acquired dendritic tree morphologies approximated as tree-structured collections of frusta and cylinders. However, they are also able to simulate simplified integrate-and-fire cells, and can combine both in hybrid networks. These simulators support multiple integration techniques, allowing different numerical methods to be tested with a single simulation to check for robustness and numerical reproducibility. All of these simulators provide efficient numerical integration of voltage or diffusion across tree-like structures [52] and support parallel network simulations [53]–[55]. NEURON also supports parallelization of an individual neuron [56]. All of these simulators utilize Python as a programming language. NeuroConstruct [57] can convert declarative NeuroML [10] models to the native formats for all of these simulators, encouraging testing of reproducibility.

B. Reaction-diffusion simulators

The lowest level in computational systems biology is molecular dynamics, but this scale is not yet much used in computational neuroscience. Reaction-diffusion covers the smallest scales that are commonly explored. Reaction-diffusion questions arise over scales that range from calcium microdomains of nanometers [58], up to calcium waves that propagate for 10s or 100s of microns down an apical dendrite [59], [60]. The simulators specialized for this domain generally only scale up to subcellular scale and rarely to whole-cell scale.

MCell [61] and Smoldyn [62] support meshless 3D particle-based stochastic simulations of small parts of a neuron. For increased performance, Smoldyn supports GPU simulation [63], [64]. Smoldyn has been integrated with MOOSE [46] andVirtual Cell (VCell) [65]. VCell [39] is widely used to simulate reaction-diffusion in computational systems biology and has also been used in neuroscience [66], [67]. The VCell simulator performs stochastic and deterministic simulations. VCell can import and export SBML reaction specifications [68] which then can be reproduced with other tools. STEPS [33], NeuroRD [69], and PSICS [41] provide stochastic reaction-diffusion simulations on relatively large morphologies corresponding to neuroanatomical morphologies [70], PSICS has partial support for the NeuroML model interchange format.

C. Large scale network simulators

Several software tools focus on simulation of large scale networks, with various approaches to this task. NEST [71] is developed for simulation of large networks of spiking neurons that can be described with just a few differential equations. Nengo is a simulation tool with graphical and scripting interfaces that allows users to generate and simulate networks of simple neurons with desired overall population behavior [34]. This simulator reaches for the highest scales and is used for studies to relate neuroscience to cognitive science and psychology. Nengo was used to create Spaun (Semantic Pointer Architecure Unified Network) [72], a 2.5-million-neuron simulation at whole-brain scale that performed a variety of behavioral tasks. The MIIND simulator [73] focuses on population level models, mainly with population density techniques. Topographica [74] is a large-scale neural model simulator with dedicated support for modeling topographic maps. The central object of neural simulation in Topographica is a Sheet, which is a two-dimensional population of neurons

with similar properties, and a typical model contains a number of sheets and connections between them. Topographica interfaces with NEST and NEURON [75].

D. Mathematically-oriented simulators

One widely used class of simulators describes their models explicitly in mathematical notation [76]. XPPAUT [77], PyDSTool [78], and Brian [30] are prominent members of this class. Equations for XPPAUT are specified by a text file, but it is otherwise controlled through a GUI interface. Although XPPAUT provides no scripting capability, it is widely used for its simplicity and fast solvers. PyDSTool is a Python module providing similar capabilities. Unlike all the other tools discussed here, PyDSTool and XPPAUT provide explicit support for bifurcation analysis. Brian describes equations for an individual neuron in an XPPAUT-like equation format as a Python string. Brian then uses Python commands to construct groups of such neurons and connect them. Numerical integration in Brian is performed using C++ or graphics processing unit (GPU)-based generated code via GeNN [14], [79]. The latter strategy is advantageous because even low end GPUs typically support many more parallel calculation pipelines than CPUs, thereby offering an inexpensive route to high speed simulations.

E. Multi-simulator tools

It is sometimes advantageous to use simulators indirectly or as a group. PyNN [32] provides a standard Python API (Application Programming Interface) for specifying models in multiple simulators. A simulation so specified may then be reproduced in other simulators to check for reproducibility and robustness. NeuroConstruct provides a GUI for constructing models, which can then be run with any of several simulators to check reproducibility [57]. NeuroConstruct also outputs a simulator-independent declarative model specification (Section IV). MUSIC is a multi-simulator standard for exchanging neuron spike events that can run parts of one simulation in multiple simulators including NEST, MOOSE, NeuroRD, and NEURON [80]. It thus allows a model to be built that takes advantage of the strengths of each simulator, encouraging building of hybrid simulations that may include different simulation parts at different levels of detail [48]. Hybrid simulations allow different simulators to be swapped in for different parts, checking reproducibility. Geppetto (geppetto.org) can be used to provide a consistent web interface for running computational neuroscience models, and is currently used by the OpenWorm project (openworm.org) [81].

III. Shared computational resources: Hardware sharing and virtualization

Lack of access to suitable hardware or software is a basic impediment to replication and reproduction. There are a number of sites and tools that address this problem.

A. The NeuroScience Gateway – shared HPC

The NeuroScience Gateway (NSG; nsgportal.org) provides free and easy access to high performance computers (HPCs) [82], [83]. It thereby reduces barriers to replicability and reproducibility for larger models that require HPCs either for simulation or analysis [84]. In addition to providing NSF-supported cycle time on these expensive large machines, NSG also makes it much easier to launch a job by providing a highly simplified web interface (the portal) that hides the complicated details needed for setting up parallel simulations as batch jobs using Torque, PBS or SLURM [85], [86]. NSG provides support for models running under several major neuroscience simulators – Brian, MOOSE, NEST, NEURON, pGENESIS (parallel GENESIS), and PyNN [53], [82]. Generally, parallel models from ModelDB can be run on NSG by uploading the zip file from ModelDB and setting a few options (*e.g.,* simulator, number of cores, name of main file).

B. Simulation virtualization

The Simulation Platform [87] was developed by the Japanese International Neuroinformatics Coordinating Facility Node (INCF Japan Node) to permit models to be run remotely through a browser on their hardware. It provides web-based access to a virtual machine to test pre-configured models from the ModelDB [44] and Visiome [88] repositories. This allows visitors to these repositories to click a *Run* button on a supported model and launch a model that will run immediately on their screen. The user can then run a simulation without having to install the simulator and related software and the model itself to run the model locally. Files generated during the session are offered for download at the end of the simulation to allow for local analysis.

NeuroDebian provides a stand-alone virtual machine that can be installed and run locally on all major platforms to immediately provide access to several neuroscience-related packages, including the simulators Brian, PyNN, and XPPAUT [89]. It also installs a set of Python modules that facilitate numerical calculations and databasing. As opposed to the INCF Japan Node Simulation Platform, the virtualization used here runs on a user's local machine and is not subject to network latency.

The virtual machine provided by NeuroDebian is based on a Debian-derived Linux OS. The packages which form part of NeuroDebian can be easily installed in other Linux systems. In this context, NeuroDebian provides a simple one-stop single source to get rapid access to a large set of tools that will then be installed directly on a user's machine and will not require any virtualization.

IV. Declarative model descriptions

Declarative model descriptions provide a formal mathematical specification of a model in a way that is impossible in natural languages. This level of description is above the computer code level in that it describes an idealized model, independent of the approximations that are necessary to simulate the model numerically. These descriptions typically provide the ability to include extensive annotations that link the mathematics to the

biology (Section V-C). Tools that can directly read and write multiple declarative descriptions, like jNeuroML [90], expand the utility of these descriptions.

Declarative model descriptions have the advantage of being useable with multiple simulators [91]. Analysis can be done on data structures used by a particular simulator but these data structures are simulator-specific and vary greatly in interpretability [23]. Simulator-independent models facilitate collaborative model development as they allow different teams to each work with their preferred tools [92]. Use of multiple simulators reduces the risk of artifacts due to simulator bugs or inappropriate usage and confirms reproducibility [93]. Standardized descriptions also allow the underlying models to be compared and searched to identify models matching specific, previously untested criteria [21].

NeuronUnit is a tool to compare models and determine consistency with experiment by rigorous comparison with experimental results [94]. NeuronUnit is thus far primarily focused on ion channels and single neuron models.

The procedural code (imperative language) used on a computer can do anything the computer is capable of doing, bugs and all. A well-designed declarative specification by contrast is limited to what is allowed by the specification – there is a *specification design* that defines what sorts of operations and functions are permitted. Generally, a specification design is created by looking bottom-up at the biology to determine what kind of natural objects are out there that need to be specified, while also looking top-down to existing models to see what objects, natural or approximations to the natural, have been typically used. When looking at existing simulators, one can pick and choose features, or attempt to find and use all features (union of existing simulator features), or use features that are commonly used (intersection of existing simulator features) [95].

Older specifications like SWC typically used a custom format [96], [97]. Newer standards are now typically developed under XML [98], or under Javascript Object Notation (JSON), because parsers for these formats are readily available for many programming languages. The COMBINE initiative (COmputational Modeling in BIology NEtwork) seeks to allow standards communities to learn from each other's designs and avoid duplication of work [99]. Although *big science* has become more common, most research is still being done by individual labs [100]. Individual labs or groups may standardize their own model formats. For example, the Cell Types database at the Allen Brain Institute (celltypes.brain-map.org) [101], provides NEURON models in a standardized JSON format runnable via their Software Development Kit (SDK). That said, the benefits of standardization can only be achieved with the involvement of the community.

Computational studies may take partial advantage of declarative specifications without using them throughout their workflow. For example, it is common for models to specify cell morphologies in the declarative SWC format, while all other descriptive levels are coded procedurally for a simulator. Declarative specifications may also be generated, automatically or by hand from a simulation model, in order to use the declarative specification as an exchange format [102]. NEURON, for example, provides the ability to export morphologies into NeuroML, regardless of how they were originally instantiated.

A. SWC morphology format

The SWC format is one of the earliest widely-used standards in computational neuroscience [96]. SWC focuses solely on describing the morphology of a neuron. In comparison to the other formats described here, SWC files have a simple encoding: each line contains several space-separated values whose meaning is interpreted based on position. These values encode a point identifier, a section type (soma, apical dendrite, distal dendrite, *etc.*), x, y, z, radius, and parent identifier (used to describe the tree structure). SWC files are widely available and widely supported: SWC is the morphology format for BigNeuron, a project to advance the state of the art for 3D neuron reconstruction [103]. All morphologies submitted to the NeuroMorpho.Org repository [97] are converted into SWC format. Some existing morphology tracing tools, such as NeuronStudio [104] and Neuromantic [105] allow saving morphologies directly into SWC format. Tools such as NLMorphologyConverter (neuronland.org) and NeuroConstruct [57] can convert SWC files to native formats suitable for use with simulators. NEURON's Import3D tool allows importing SWC morphologies directly without an intermediate translation step.

B. NeuroML and NineML

NeuroML [10], [21] is an XML-based neuroscience-specific standard designed for the specification of computational neuroscience models. NeuroML is serializable as either XML or JSON [106]. In NeuroML version 1, channel kinetics was limited to a set of predefined forms; version 2, currently in beta, introduces Low Entropy Model Specification (LEMS) [90], which allows modelers to define their own kinetic forms. It continues to provide a reference set of channel types [106]. To reduce the risk of misinterpreting models, the NeuroML standard requires all units to be explicitly specified [106]. The NeuroML group provides a number of tools to simplify working with NeuroML, including libNeuroML and PyLEMS [106]. Many simulators and other analysis tools provide at least partial support for NeuroML (a list is provided at www.neuroml.org/tool_support). Additional simulator support is available by converting NeuroML models to PyNN [32] via NeuroConstruct [107].

Where NeuroML is oriented toward morphologically detailed models of single cells or networks of these cells, NineML (Network Interchange for Neuroscience Modeling Language) – developed by the INCF – focuses more on large networks of integrate-and-fire type neurons [6]. In NineML the levels of description are not physical scales but are more conceptual, with an *abstraction layer* and a *user layer* [108].

C. SBML, CellML, SED-ML

SBML, CellML [109], [110] and SED-ML (Simulation Experiment Description Markup Language) [111] are core standards of the COMBINE initiative [99]. SBML is an

XML-based model description widely used in Computational Systems Biology, which was introduced to promote model exchange and accessibility [68]. SBML enjoys the support of more than 280 simulation environments (sbml.org/SBML_Software_Guide). This makes it possible to check reproducibility by comparing results of computations performed by several different simulators for the same model [112].

The basic SBML components are species and reactions. Despite this, its applications are not limited to biochemistry. SBML species can be used to represent higher-level entities such as a cell, organ or organism, with kinetic rate rules then used to describe continuous change of any quantitative parameter at any of these scales, and events used to describe any discontinuous change. There exists a number of neuroscience related models in SBML format, as well as models that are of general interest in biology. Researchers have also developed models that include both Computational Neuroscience and Systems Biology components, creating, for example, a multiscale simulation of a neuron or network with subcomponents modified from SBML models and other sources [51], [113]. SBMLMerge assists with merging separate SBML files into a combined model [114]. Importantly, a new set of features in the SBML Level 3 Spatial Processes package under development will include spatial aspects of molecular distributions. This will be of great value for building into neuronal models where the "well mixed" assumption of current SBML is not generally valid.

Among neuroscience simulation software, SBML is supported in whole or in part by STEPS, Moose, NEURON, VCell, XPPAUT, and jNeuroML. LibSBML provides APIs for working with SBML from Python and other programming languages, giving access from other neural simulation programs that use Python as the interpreter. JSBML offers a pure Java alternative. In addition, it is possible to convert SBML to LEMS, so any tool that can import LEMS can use converted SBML models [90].

CellML, often used for the same class of problems as SBML, adopts a different philosophical and structural approach. SBML files are hierarchical and encode biological information in SBML tags. CellML files are relatively flat and describe the model as a collection of mathematical components (*e.g.,* equations describing an IP3R) with associated dynamics and optional semantic biological metadata in RDF (Resource Description Framework) [99], [102]. Like SBML, CellML reaction dynamics are specified in a subset of MathML [115], although CellML requires units to be explicitly specified [102] whereas SBML has default units. In CellML, parameter values do not need to be specified, however. CellML uses this feature to allow partial model descriptions for these qualitative models [102]. An official CellML API [116] accessible via C++, Java, and Python, offers support for reading, writing, and generating code from CellML files. CellML models are non-spatial. Nonetheless, CellML has been used for a number of neuroscience models. FieldML [117], [118] has been proposed [102] as a way to spatially extend CellML models. CellML repository software now supports visualizing FieldML via the Zinc plugin [119]. Despite their differences, SBML and CellML are largely interconvertible by tools like Antimony [120].

SED-ML was developed for exchange of descriptions of any simulation experiment but has thus far been primarily used in computational biology. A SED-ML file can contain references to re-used models, pre-processing procedures, information about the simulation steps and settings, post-processing procedures, and specifications of simulation output such as what plots to create. It can specify the simulation parameters required for models described in SBML, NeuroML, VCML or CellML. Pre- and post-processing can be specified using MathML [115]. Simulation algorithms can be further specified using Kinetic Simulation Algorithm Ontology KiSAO [121]. SED-ML encourages reproducibility by allowing model authors to complete the description of how their simulations should be run. After making changes to simulation, a user can document their simulation experiment by exporting the modified SED-ML description [111].

V. Repositories and standards: code, data and models

Sharing model code is essential for replicability of a simulation by others. Although code is nominally not needed for reproducibility, which is by definition independent of implementation, shared code is typically needed to provide model details which are under-defined by the equations provided in a journal article – *e.g.,* the choice of numerical integration method. Software also facilitates reuse, the form of reproduction that implicitly validates the old result while adding new information [9].

Simulation data sharing is valuable for determining that replication is precise and whether reproduction is precise, close, or if it failed. However simulation data is in many cases extremely large and is therefore typically not made available.

Whether a model has been defined declaratively in a descriptive language like NeuroML, or programmatically through simulator code, an electronic copy is both more accurate and more immediately useful than a paper copy. Even when the paper version is complete and entirely without error, errors are likely to creep in when it is typed or scanned to get it back into a computer.

Explicit policies and standards can be set at several different levels of research assessment: 1. funding agencies; 2. publishers; 3. repositories. In the clinical realm, a consortium of major journals recently proposed requiring full data sharing for all clinical articles published [122]. Although appropriate policy is important, it is the culture of a community that dictates what people actually do. In computational neuroscience, most practitioners have embraced the culture of sharing, though some resist it [123]. A few journals, or individual editors, require that software be shared as a condition of publication, noting that reproduction is not practical in the absence of code. Additionally, federal funding agencies which support much of this research are also encouraging sharing, and require sharing as a condition of support for some funding opportunities. One method to change the culture is to list all of the potential models for a repository, whether shared

or not. Notifying authors that their work would be listed as "unavailable" was enough to encourage many to provide files for the NeuroMorpho.Org repository [124].

The rigor of policies must be balanced against the difficulty of following them – if policies are too onerous or sharing too difficult, investigators may avoid sharing or will game the system and only comply with them nominally. In some cases, guidance and guidelines rather than requirements may be more effective.

Model sharing can take many forms [6]. At the simplest level, a researcher can share source code files via email when someone requests them. On-demand sharing is effective, but it is potentially risky since the original files have to be located. Finding old files reliably requires that one uses a version control system systematically. Otherwise, files may get deleted or modified. In the common case of code written by a trainee, the files may not even be known to the people remaining in the laboratory.

As a safeguard against such risks, many researchers have established lab policies for sharing code. Many place code on the research group's website. This strategy has been adopted by both individual researchers and large research groups, such as the Allen Institute's Cell Types Database (celltypes.brain-map.org) and the Human Brain Project's Neocortical Micro-circuit Collaboration Portal (bbp.epfl.ch/nmc-portal/welcome) [125]. However, especially for small groups, lab websites are often only maintained as long as the lead researcher continues to work at the same university [126]. A further difficulty with the private-site approach is the difficulty of finding models that are not deposited in any central location. Models are only successfully shared to the extent that they are *discoverable* [6]. An alternative to private sites is to share on a public repository like GitHub (GitHub.com), which has the advantage of providing version control as well as public access. Use of a public repository also means that the code will continue to be available even if its laboratory of origin moves or is closed. GitHub is used, for example, for sharing code by the OpenWorm project (GitHub.com/openworm) [81].

Making models more discoverable promotes new collaborations and model re-use [127]. Sharing a set of models in a central repository greatly improves discoverability compared to having the same models available across multiple individual lab websites [128]. To address the problem of discoverability, the computational neuroscience community has developed several specialized databases for model sharing. Having multiple models in one place also enables ready feature comparison across models. In addition to providing a common point-of-entry to find models from many labs, many of these databases provide some levels of quality control [6]. Use of version-control in a repository allows sharing of bugfixes and other improvements [6], while still providing access to an earlier, published, canonical model version. Shared models also provide working examples of various techniques and problem solutions that are valuable for neuroscience education [129].

A. Specific sharing sites

1) ModelDB: The ModelDB repository (modeldb.yale.edu) was developed in 1996 to allow model sharing and facilitate model re-use and collaboration. By policy, ModelDB only makes a model public after it has been used in a peer-reviewed publication; the publication serves as background and partial documentation for understanding the simulation and interpreting the code. Each model continues to be manually curated to ensure that it replicates a figure in a published paper, and to make minor changes where required to allow the model to run on the 3 major platforms (Windows, Mac, and Linux) where possible. ModelDB has grown to contain over 1000 public models coded in over 70 different languages, simulators and simulation environments. (Many simulations are coded directly in a general-purpose programming language and not in a simulator). However, there remains problems convincing people to submit their models. For example NEURON has been used in over 1500 publications (list at neuron.yale.edu/neuron/static/bib/usednrn.html), but only about one third of these models available on ModelDB. Brian has been used in over 100 publications, but only one tenth of these are available on ModelDB.

Easy model deposition encourages sharing. Generally, an investigator can upload their running code to ModelDB without change – ModelDB does not require any model conversion or standard model description format. Since 2004, web forms were added to provide baseline model characterization at submission through menus of possible descriptors such as cell type, brain region, topic, *etc.* [130]. These descriptors are augmented and, if necessary, revised by the curator. If a *README* providing instructions for running the simulation is not provided by the investigator, it is added by the curator. Although there are no particular rules about commenting or documentation, many investigators provide fairly clean code, realizing that deposited models will often be viewed together with their publications. A Mercurial (mercurial-scm.org) repository with revision history is provided for some models.

Models may be located in ModelDB by searching for an author, by searching the text of the model, or by searching or browsing by the curated metadata. Additional descriptors cover 129 topics across a broad range of subject areas: disease states such as Alzheimers, learning and behavioral algorithms such as pattern recognition, physiological measures such as ion channel kinetics, and others. ModelDB's ModelView [23] provides an additional tool for examining NEURON models on the web before downloading or running them. ModelView automatically parses code to provide a structured view into a model, describing, for example, cell morphologies, distributions of conductances, numbers of cells, network structure.

2) OpenSourceBrain: OpenSourceBrain (OSB) [131] (opensourcebrain.org) promotes collaborative development of unpublished work, as well as providing the potential for new or continued collaboration for work that has already been published. OSB provides version control for all deposited models, facilitating groups working together. Models can be submitted in any language, with OSB providing different

levels of support for different languages, with particular support for NEURON, GENESIS, and MOOSE, and a focus on NeuroML as a higher level descriptive language. OSB provides strong model visualization for models described in NeuroML. Each project on OSB is linked to an individual public version control repository where the model code is shared, usually on GitHub. Researchers can use OSB tools to see who else is developing a given model and what changes others have made. Once a model is published, OSB also links to the model copy on ModelDB.

3) NeuroMorpho.Org: NeuroMorpho.Org [97] is a specialized repository for morphological reconstructions; its version 6.2 release (October 6, 2015) contains 34,082 reconstructed neurons from 165 cell types. Each reconstruction is tagged with metadata including species, gender, age, and staining method. This database distributes both the original files and a version standardized into the SWC format [96]. Their 3D viewer tool additionally allows morphologies to be exported into NeuroML, native NEURON, and native GENESIS formats. Every month, NeuroMorpho.Org contacts authors of all papers with new morphologies identified by a literature survey to request that they share their data. The full list of papers with known morphologies and whether or not they are available in NeuroMorpho.Org is released on the repository's website [132].

4) Channelpedia: Channelpedia (channelpedia.epfl.ch) is another specialized resource; it specializes in aggregating information about ion channels with over 50 published models available, relevant to both neuroscience and cardiac modeling. Much of the information (*e.g.,* location of the coding genes in various species) is not yet relevant for the current level of computational neuroscience research. Channelpedia distributes ion channel models in several formats, including NMODL for NEURON [133], [134] and ChannelML, a subset of NeuroML [10]. Channelpedia provides figures showing the response of the channel model and gating dynamics to voltage clamp conditions, allowing the models to be validated by comparison with experimental data.

5) Other repositories: A number of general model repositories also house neuroscience models.

The CellML repository [135] (models.cellml.org) is powered by Physiome Model Repository 2 software [136] and stores both neuroscience and non-neuroscience models as long as they are expressed in CellML [109]. There is a separate category for Neurobiology (33 models at this time), but neuroscience simulations are also found under additional categories (203 found with search term "neuro") including electrophysiology, calcium dynamics, and circadian rhythms. The Physiome software provides the ability to view a model in several ways: as raw CellML, equations, or converted to other formats.

The Visiome repository [88] (visiome.neuroinf.jp) hosts 83 published models, in addition to other vision-related resources. These models work with a variety of tools including MATLAB and NEURON. One feature of Visiome is that it provides an explicit statement of license rights to each model's code, which appears on the model's display page. Another interesting feature is to identify the natural language, mostly English and Japanese, used in commenting and documentation for a particular model.

The BioModels database (biomodels.org) is a repository containing hundreds of published models converted to SBML [137]. It has multiple models related to neuroscience including models of spiking neurons. Annotations connect model elements to records in external database resources and allow efficient search, comparison, and merging of models. For curated models, the website provides a user interface for exploring model elements, downloading a sub-model with selected elements, viewing curator comments and figures representing recalculated results corresponding to figures in original publications, running simulations online, *etc.*

Much of the information available in the various databases described here is searchable through a single website: the Neuroscience Information Framework (NIF; neuinfo.org) [138], [139]. Information located by searching this data federation is linked to its database of origin, which may offer additional visualization and analysis tools.

B. Sharing simulation output

Although most often thought about in the context of experimental science, sharing data – the output of simulations – also promotes reproducibility in computation, providing direct reference values for reproducibility comparison. This is particularly important if the original model is not replicable due to code or tool availability issues. Some model repositories, such as JWS Online, address this challenge by running models on demand to regenerate data [140]. This approach works for small, quickly runnable models. However, computational neuroscience models can generate a vast amount of data and this data can grow without limit depending on how much parameter exploration is done or how long the simulations are run for.

Although there are still no widely-accepted standards for data format or distribution for simulation data [141], some preliminary standards have been proposed. The Neuroscience Simulation Data Format (NSDF; GitHub.com/nsdf/nsdf) [142] is specifically designed for storing simulation data as opposed to experimental data. A high-level API allows accessing the simulation data from the underlying HDF5 database. An alternative data storage approach is to reuse experimental data storage formats, although the experimental community also lacks standards. The Neurodata Without Borders framework for data sharing is a collaborative project involving several large research groups [143]. NIX (GitHub.com/G-Node/nix) is an alternative format under development by the INCF German Node. All three of these formats embed their data in a Hierarchical Data Format (HDF5) [144], which allows third-party tools to work with the files using existing HDF5 readers.

C. Model annotation

Models must be interpretable to be useful. Ideally, models should be interpretable as neuroscience and readily explicable to a neuroscientist who is not a modeler. Therefore, the model should be more than just a set of equations or subroutines –

comments and annotations should provide additional information [124], [127], [129], [145].

Some model repositories provide annotation as part of the curation process. For example, ModelDB [44] provides annotation in the form of accompanying metadata for each model. Curated BioModels [146] entries have the annotations embedded in the SBML files themselves. These annotations typically map model parameters and equations to biological concepts, but they may also map the model itself to the dataset that inspired it [147]. Related information may or may not be included in a model's associated publication, but putting all of the annotation in a machine readable form allows tools like the NIF LinkOut Broker [148] to automatically find complementary information and resources.

For maximum efficacy, a consistent terminology is necessary for the annotations. The NeuroLex ontology [149], [150] (neurolex.org), powered by the NIF, provides a hierarchically structured set of neuroscience terms. Some external resources (e.g. Open Source Brain) and journal articles use this ontology to provide an unambiguous identification of the neurons, etc involved. Another NIF-affiliated project, the Resource Identification Initiative (RII) provides similar standardized Research Resource Identifiers (RRIDs) for organisms, antibodies, software tools, and databases. Dozens of journal editors have partnered with RII to promote the use of RRIDs [151]. The Mathematical Modeling Ontology (bioportal.bioontology.org/ontologies/MAMO) provides standard terms for classifying categories of models (*e.g.,* MAMO_0000045 : *differential equation model* is a child of MAMO_0000003 : *mathematical model*). A preliminary version of a Computational Neuroscience Ontology (CNO) has also been developed [152]. CNO reuses terms from the Systems Biology Ontology [121] and elsewhere where appropriate.

D. Model sharing standards and policies

Many of the tools and repositories discussed in this paper provide and encourage use of certain standards for model annotation and sharing activities. At a higher level of definition, some proposals have sought to identify aspects of model definition that should be enforced by *any* repository or model definition standard – standards for the standards. For example, one extensive proposal for information annotation standards addressed: hypothesis, model derivation, model description, implementation details, analysis, and supporting evidence [128]. Another proposal suggested that all individual figures in a publication should carry necessary meta-data for critical information via version numbers of specific parameters [6].

The Minimum Information About a Simulation Experiment (MIASE) [153] standards proposed ten sharing rules that should ensure reproducibility. The rules fall under three categories which can be viewed as a minimal basis for sharing: 1. describe the model including all parameters and equations; 2. describe the simulation process, both the virtual experiment and numerical methods 3. describe data analysis and how raw data is transformed into results presented. At the descriptive level, these could be satisfied by using a NeuroML description with SED-ML for categories 2 and 3. MIASE standards can also be satisfied by sharing code, assuming adequate code transparency. For example, making sure to separate model description from simulation procedure maintains the distinction between the above categories, thereby improving clarity and reproducibility [147].

In addition to repository standards, model sharing and reproducibility can be mandated by the setting of policies by organizations in positions of power, namely publishers and funding agencies. In the United States, the National Institutes of Health (NIH) have data sharing mandates in place for certain types of grants and for grants of large size, but these mandates are not clearly spelled out in the case of computer simulation and serve more as encouragement rather than requirement. Some specific modeling-oriented NIH and National Science Foundation (NSF) funding initiatives have specific model sharing requirements built in to the request for proposals. Recently, the NIH has begun development of 4 focus areas for a new *rigor and transparency* policy (grants.nih.gov/reproducibility/index.htm). The third of these includes "full transparency in reporting experimental details so that others may reproduce and extend the findings." On the publication side, the PLOS journals and Nature Methods require that a model be shared when it forms a key part of the work – some papers may include models that are trivial, standard, or are incidental to the point of the paper. Similarly, other journals encourage the individual editor to require model sharing on a case-by-case basis for those papers that are primarily model-based.

E. Provenance and version control

Understanding the *provenance* of a model, or of a specific result, greatly improves scientific usefulness. Provenance for a particular model will ideally include details of what paper or what experiment was used to determine a particular model result [154]. Provenance for a particular result includes what data transformations were needed to go from raw data to a figure or table. In computation, result provenance can be provided by an algorithm at the descriptive level, or by a script that produces the result, up to the point of putting up a particular figure from a paper. *Version control* is the standard and best method of providing an unbroken provenance chain for software, but can also be used for other documents related to the research. In experimental research, the laboratory notebook is traditionally regarded as the single site for determining a result's provenance and documenting that an experiment reported was actually done. In modeling, a *electronic notebook* (enotebook) is typically used instead of paper documents, allowing them to be an integral part of the research process. Placing a daily-dated electronic notebook under version control with a daily check-in makes it much more difficult to forge notebook entries at a later date. Integrating notebooks with version control complements a version control system's own comment chain and preserves information about what results were generated when, with which parameters, and with which version of the code.

Internal reproducibility can be further enhanced by adherence to standard workflows and by implementing these as much as possible through *workflow automation* with tools like Mozaik [155] and Lancet [156]. Similarly, databases associated with parameter provenance and other research documents can also be added to one or more repositories associated with a research project. In this way, a later researcher can follow the full chain of what has been done, being able to note, for example, when and why a particular parameter was changed and being able to reproduce a model with either version of that parameter. In some cases, simpler initial models saved from an earlier stage in the development process may offer greater conceptual clarity due to their simplicity.

Collaborative development, whether across laboratories or within one laboratory, greatly benefits from version control, permitting each investigator to reproduce the work of the other and to immediately see differences in code or parameters. Modelers may version locally with standard version control software like Subversion, Git or Mercurial or on a general-purpose public repository like GitHub. OpenSourceBrain promotes [131] collaborative development backed by GitHub and other public version control repositories. The ModelDB repository allows but does not require Mercurial-based versioning. Sumatra provides a simulator independent method to record a command-line and code version used to run a simulation [157], [158]. SBML allows model history to be embedded in the SBML file itself [159]. Some integrated development environments (IDEs) for Python provide similar facilities.

F. Licensing

Law school textbooks could and surely will be written about the issues of copyright and licensing in biomedical simulation, a subtopic within the laws of software that must also take into account the likelihood that such simulations will eventually be used on people in a clinical environment. Clinical applications will require far more attention to credibility, prior disclosure, and malpractice protection than is required for models developed for other modelers (see torte law).

Sharing a model does not necessarily grant immediate legal rights for others to use it [92]. Therefore license specification is important. Specification may be specified in free text, ideally using standard license wording provided by major open-source licensing agreements such as GPL, MIT and BSD licensing documents. GPL is somewhat stricter than the other two since it requires that commercial uses release any modified open-source code as also open-source (called a copyleft requirement). Federal funding agencies have in some cases requested that software developed under their auspices be open-source and permit subsequent commercial use without such restrictions. In addition to textual form, a machine-readable form of license may also be useful [100]. When code is obtained via a repository, the repository may itself add additional conditions for reuse. Typically, a repository will require that the repository and the original author both be cited [127].

VI. Broader issues – experimental data sharing and model matching

Ultimately, reproducibility in science must unify models and experiments. Experimental data is critical both for constraining computational model parameters and for evaluating the output of models. There is an enormous amount of detailed information out there in the experimental community, but it is difficult to extract from papers, and not yet generally available in shared databases. As *big data* becomes a bigger part of neuroscience, there is increasing emphasis on providing centralized data-basing of experimental data. This brings up a large number of additional issues that are beyond the scope of this article, particularly with respect to determining appropriate metadata for physiological experiments performed under different conditions and, in the case of *in vivo* experiments, with completely different tasks.

Major efforts in experimental database are being developed under the auspices of the Human Brain Project (humanbrainproject.eu), Allen Brain Atlas (brain-map.org) and in projects associated with the US Brain Research through Advancing Innovative Neurotechnologies (braininitiative.nih.gov). Additional specific efforts for experimental data sharing include Collaborative Research in Computational Neuroscience [141] (crcns.org) hosting Neurodata Without Borders [143], Hippocampome [160] (hippocampome.org), and NeuroElectro [161] (neuroelectro.org).

Even as additional quantitative information pours in, experimental data for parameters will invariably remain incomplete. Therefore, there will be a need to add to these databases, or to their modeling affiliates, explicit lists of assumptions, and alternative assumptions, about missing parameters [154]. These could then be accessed by experimentalists to provide a wish-list of additional measurements to pursue. Some of these measurements will not be possible at present, but will become possible as new innovative recording and imaging neurotechnologies are developed. For example, it may eventually be necessary to obtain point-to-point wiring data from individual animals in order to determine general rules for how cortical circuitry works.

As we develop simulation technology, we must address critical gaps between simulation and reality. Just as an *in vitro* model is only of value if it reproduces an *in vivo* reality, simulation must make predictions that can be confirmed experimentally. While such a confirmation enhances our trust in a model, it cannot show the model to be correct or complete [162]. Meanwhile, failure of confirmation may or may not refute, or falsify, a model, depending on the degree of reproducibility of the experiment and phenomenon being modeled (*cf.* "overdispersion" in place cells [11]). In physics modeling, a falsified model is discarded. In biological modeling, as in weather modeling, a falsified model is updated and retried.

VII. Conclusions

At the level of individual investigator, best practice in model sharing and reproducibility is comparable to best practice in other software development:

- *Version control.* Free cloud-based version control services now makes version control readily available. Version control is particularly important during model development where it enhances productivity. Additionally, a simulation typically goes through many variants to create different figures – version identifiers associated with particular figures make it possible to replicate individual figures in a paper.
- *Strong commenting and documentation.* The journal article associated with the model often serves as the model's primary documentation, and should have text and labels aligned with code comments and ancillary code documentation. This should include ontology-based (*e.g.,* NeuroLex) annotations for biological concepts, algorithms, and simulator tools. Experimental data or other sources of parameter values should be fully cited, with gaps in parameter sourcing ideally fully acknowledged.
- *Code modularity.* Modular coding facilitates testing, promotes reuse, and reduces the risk of implementation errors. Code reused from other models should be cited both in the code and in the paper.
- *Open source sharing.* Submitting models to online repositories like ModelDB not only benefits the community but also benefits the user by leading to increased visibility and citations.

At the community level, repositories can take action by encouraging the submission of structured metadata, and by providing tools for examining models. Journals and funding agencies can promote reproducibility by requiring model sharing as a condition for publishing or for obtaining funding.

Development of standards for reproducibility will ensure reproducibility as major models of specific brain areas grow beyond the bounds of management by a single laboratory. Complex models of neocortex, hippocampus, cerebellum, *etc.* will reach a level of sophistication where their use and development is shared among laboratories, individual labs using these established models in their studies.

ACKNOWLEDGMENT

The authors would like to thank the Shepherd lab at Yale University and the Neurosim lab at SUNY Downstate, and the Interagency Modeling and Analysis Group Multiscale Modeling meetings for presentations and conversations that informed the development of this manuscript. Supported by NIH grants T15LM007056, R01MH086638, and U01EB017695.

REFERENCES

[1] N. Toni *et al.*, "Synapse formation on neurons born in the adult hippocampus," *Nature Neuroscience*, vol. 10, no. 6, pp. 727–734, 2007.
[2] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," *The Journal of Physiology*, vol. 117, no. 4, pp. 500–544, 1952.
[3] A. Baddeley, "Working memory," *Science*, vol. 255, no. 5044, pp. 556–559, 1992.
[4] L. Lapicque, "Recherches quantitatives sur l'excitation électrique des nerfs traitée comme une polarisation," *J. Physiol. Pathol. Gen*, vol. 9, no. 1, pp. 620–635, 1907.
[5] C. Drummond, "Replicability is not reproducibility: nor is it good science," 2009.
[6] S. M. Crook *et al.*, "Learning from the past: approaches for reproducibility in computational neuroscience," in *20 Years of Computational Neuroscience*. Springer, 2013, pp. 73–102.
[7] R. Williams, "Can't get no reproduction: leading researchers discuss the problem of irreproducible results," *Circ Res*, vol. 117, pp. 667–670, 2015.
[8] D. Castelvecchi, "The biggest mystery in mathematics: Shinichi Mochizuki and the impenetrable proof," *Nature*, vol. 526, pp. 178–181, 2015.
[9] A. Casadevall and F. C. Fang, "Reproducible science," *Infection and Immunity*, vol. 78, no. 12, pp. 4972–4975, 2010.
[10] P. Gleeson *et al.*, "NeuroML: a language for describing data driven models of neurons and networks with a high degree of biological detail." *PLoS Computational Biology*, vol. 6, no. 6, p. e1000815, 2010.
[11] A. Fenton *et al.*, "Attention-like modulation of hippocampus place cell discharge," *J Neurosci*, vol. 30, pp. 4613–4625, 2010.
[12] S. S. Mukherjee *et al.*, "The soft error problem: An architectural perspective," in *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*. IEEE, 2005, pp. 243–247.
[13] R. Ben-Shalom *et al.*, "Accelerating compartmental modeling on a graphical processing unit," *Frontiers in Neuroinformatics*, vol. 7, 2013.
[14] E. Yavuz *et al.*, "GeNN: a code generation framework for accelerated brain simulations," *Scientific Reports*, vol. 6, p. 18854, 2016.
[15] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.
[16] R. Silver *et al.*, "Neurotech for neuroscience: unifying concepts, organizing principles, and emerging tools," *The Journal of Neuroscience*, vol. 27, no. 44, pp. 11 807–11 819, 2007.
[17] M. M. Khan *et al.*, "SpiNNaker: mapping neural networks onto a massively-parallel chip multiprocessor," in *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. Ieee, 2008, pp. 2849–2856.
[18] X. Jin *et al.*, "Modeling spiking neural networks on SpiNNaker," *Computing in Science & Engineering*, vol. 12, no. 5, pp. 91–97, 2010.
[19] W. T. Weeks *et al.*, "Algorithms for ASTAP–a network-analysis program," *Circuit Theory, IEEE Transactions on*, vol. 20, no. 6, pp. 628–634, 1973.
[20] G. M. Shepherd and R. K. Brayton, "Computer simulation of a dendrodendritic synaptic circuit for self-and lateral-inhibition in the olfactory bulb," *Brain Research*, vol. 175, no. 2, pp. 377–382, 1979.
[21] N. H. Goddard *et al.*, "Towards NeuroML: model description methods for collaborative modelling in neuroscience," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 356, no. 1412, pp. 1209–1228, 2001.
[22] R. A. McDougal *et al.*, "Reaction-diffusion in the NEURON simulator," *Frontiers in Neuroinformatics*, vol. 7, 2013.
[23] ——, "ModelView for ModelDB: Online presentation of model structure," *Neuroinformatics*, pp. 1–12, 2015.
[24] A. M. Uhrmacher, "Seven pitfalls in modeling and simulation research," in *Proceedings of the Winter Simulation Conference*. Winter Simulation Conference, 2012, p. 318.
[25] J. G. King *et al.*, "A component-based extension framework for large-scale parallel simulations in NEURON," *Frontiers in Neuroinformatics*, vol. 3, 2009.
[26] G. Van Rossum and F. L. Drake Jr, *Python reference manual*. Centrum voor Wiskunde en Informatica Amsterdam, 1995.
[27] A. Davison *et al.*, "Trends in programming languages for neuroscience simulations," *Front Neurosci*, vol. 3, pp. 374–380, 2009.
[28] S. Ray and U. S. Bhalla, "PyMOOSE: interoperable scripting in Python for MOOSE," *Frontiers in Neuroinformatics*, vol. 2, no. 6, pp. 1–16, 2008.
[29] M. L. Hines *et al.*, "NEURON and Python," *Frontiers in Neuroinformatics*, vol. 3, 2009.
[30] D. Goodman and R. Brette, "Brian: a simulator for spiking neural networks in Python," *Frontiers in Neuroinformatics*, vol. 2, 2008.
[31] D. Pecevski *et al.*, "PCSIM: a parallel simulation environment for neural circuits fully integrated with Python," *Frontiers in Neuroinformatics*, vol. 3, 2009.
[32] A. P. Davison *et al.*, "PyNN: a common interface for neuronal network simulators," *Frontiers in Neuroinformatics*, vol. 2, 2008.
[33] S. Wils and E. De Schutter, "STEPS: modeling and simulating complex reaction-diffusion systems with Python," *Frontiers in Neuroinformatics*, vol. 3, 2009.
[34] T. C. Stewart *et al.*, "Python scripting in the Nengo simulator," *Frontiers in Neuroinformatics*, vol. 3, 2009.
[35] J. M. Eppler *et al.*, "PyNEST: a convenient interface to the NEST simulator," *Frontiers in Neuroinformatics*, vol. 2, 2008.

[36] H. Cornelis *et al.*, "Python as a federation tool for GENESIS 3.0," *PLoS One*, vol. 7, no. 1, p. e29018, 2012.

[37] I. Ziv *et al.*, "Simulator for neural networks and action potentials: description and application," *Journal of Neurophysiology*, vol. 71, no. 1, pp. 294–308, 1994.

[38] D. A. Baxter and J. H. Byrne, "Simulator for neural networks and action potentials," in *Neuroinformatics*. Springer, 2007, pp. 127–154.

[39] L. M. Loew and J. C. Schaff, "The Virtual Cell: a software environment for computational cell biology," *TRENDS in Biotechnology*, vol. 19, no. 10, pp. 401–406, 2001.

[40] M. Hines and N. T. Carnevale, "NEURON: a tool for neuroscientists," *The Neuroscientist*, vol. 7, no. 2, pp. 123–135, 2001.

[41] R. C. Cannon *et al.*, "Stochastic ion channel gating in dendritic neurons: Morphology dependence and probabilistic synaptic activation of dendritic spikes," *PLoS Comput Biol*, vol. 6, no. 8, p. e1000886, 08 2010. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pcbi.1000886

[42] J. M. Bower and D. Beeman, *The book of GENESIS: exploring realistic neural models with the GEneral NEural SImulation System*. Springer Science & Business Media, 2012.

[43] M. L. Hines *et al.*, "Model structure analysis in NEURON," in *Neuroinformatics*. Springer, 2007, pp. 91–102.

[44] M. Migliore *et al.*, "ModelDB," *Neuroinformatics*, vol. 1, no. 1, pp. 135–139, 2003.

[45] T. Sejnowski *et al.*, "Computational Neuroscience," *Science*, vol. 241, pp. 1299–1306, 1988.

[46] S. Ray *et al.*, "A general biological simulator: the multiscale object oriented simulation environment, MOOSE," *BMC Neuroscience*, vol. 9, no. Suppl 1, p. P93, 2008.

[47] U. S. Bhalla, "Use of Kinetikit and GENESIS for modeling signaling pathways," *Methods in Enzymology*, vol. 345, p. 3, 2002.

[48] M. Brandi *et al.*, "Connecting MOOSE and NeuroRD through MUSIC: towards a communication framework for multi-scale modeling," *BMC Neuroscience*, vol. 12, no. Suppl 1, p. P77, 2011.

[49] M. Migliore *et al.*, "Distributed organization of a brain microcircuit analyzed by three-dimensional modeling: the olfactory bulb," *Front Comput Neurosci*, vol. 8, p. 50, 2014.

[50] H. Markram *et al.*, "Reconstruction and simulation of neocortical microcircuitry," *Cell*, vol. 163, pp. 456–492, 2015.

[51] S. Neymotin *et al.*, "Calcium regulation of HCN channels supports persistent activity in a multiscale model of neocortex," *Neurosci*, vol. 316, no. 1, pp. 344–366, 2016.

[52] M. Hines, "Efficient computation of branched nerve equations," *International Journal of Bio-medical Computing*, vol. 15, no. 1, pp. 69–76, 1984.

[53] N. H. Goddard and G. Hood, "Parallel GENESIS for large-scale modeling," in *Computational Neuroscience*. Springer, 1997, pp. 911–917.

[54] M. Migliore *et al.*, "Parallel network simulations with NEURON," *Journal of Computational Neuroscience*, vol. 21, no. 2, pp. 119–129, 2006.

[55] N. Dudani *et al.*, "Multiscale modeling and interoperability in MOOSE," *BMC Neuroscience*, vol. 10, no. Suppl 1, p. P54, 2009.

[56] M. L. Hines *et al.*, "Fully implicit parallel simulation of single neurons," *Journal of Computational Neuroscience*, vol. 25, no. 3, pp. 439–448, 2008.

[57] P. Gleeson *et al.*, "neuroConstruct: a tool for modeling networks of neurons in 3D space," *Neuron*, vol. 54, no. 2, pp. 219–235, 2007.

[58] D. X. Keller *et al.*, "Calmodulin activation by calcium transients in the postsynaptic density of dendritic spines," *PLoS One*, vol. 3, no. 4, pp. e2045–e2045, 2008.

[59] B. E. Peercy, "Initiation and propagation of a neuronal intracellular calcium wave," *Journal of Computational Neuroscience*, vol. 25, no. 2, pp. 334–348, 2008.

[60] S. A. Neymotin *et al.*, "Neuronal calcium wave propagation varies with changes in endoplasmic reticulum parameters: a computer model," *Neural Computation*, 2015.

[61] J. R. Stiles *et al.*, "Monte Carlo methods for simulating realistic synaptic microphysiology using MCell," *Computational Neuroscience: Realistic Modeling for Experimentalists*, pp. 87–127, 2001.

[62] S. S. Andrews *et al.*, "Detailed simulations of cell biology with Smoldyn 2.1," *PLoS Comput Biol*, vol. 6, no. 3, p. e1000705, 2010.

[63] D. V. Gladkov *et al.*, "Accelerating the Smoldyn spatial stochastic biochemical reaction network simulator using GPUs," in *Proceedings of the 19th High Performance Computing Symposia*. Society for Computer Simulation International, 2011, pp. 151–158.

[64] L. Dematté, "Smoldyn on graphics processing units: massively parallel brownian dynamics simulations," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, vol. 9, no. 3, pp. 655–667, 2012.

[65] S.-A. Brown *et al.*, "Spatial organization and diffusion in neuronal signaling," in *Computational Systems Neurobiology*. Springer, 2012, pp. 133–161.

[66] ——, "Virtual NEURON: a strategy for merged biochemical and electrophysiological modeling," *Journal of Computational Neuroscience*, vol. 31, no. 2, pp. 385–400, 2011.

[67] N. Hernjak *et al.*, "Modeling and analysis of calcium signaling events leading to long-term depression in cerebellar Purkinje cells," *Biophysical Journal*, vol. 89, no. 6, pp. 3790–3806, 2005.

[68] M. Hucka *et al.*, "The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, no. 4, pp. 524–531, 2003.

[69] R. F. Oliveira *et al.*, "The role of type 4 phosphodiesterases in generating microdomains of cAMP: large scale stochastic simulations," *PloS One*, vol. 5, no. 7, p. e11725, 2010.

[70] J. R. Glaser and E. M. Glaser, "Neuron imaging with Neurolucida – a PC-based system for image combining microscopy," *Computerized Medical Imaging and Graphics*, vol. 14, no. 5, pp. 307–317, 1990.

[71] H. Plesser *et al.*, "NEST: the neural simulation tool," in *Encyclopedia of Computational Neuroscience*, D. Jaeger and R. Jung, Eds. Springer New York, 2015, pp. 1849–1852.

[72] C. Eliasmith *et al.*, "A large-scale model of the functioning brain," *Science*, vol. 338, no. 6111, pp. 1202–1205, 2012.

[73] M. de Kamps and V. Baier, "Multiple interacting instantiations of neuronal dynamics (MIIND): a library for rapid prototyping of models in cognitive neuroscience," in *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, Aug 2007, pp. 2829–2834.

[74] J. A. Bednar, "Understanding neural maps with Topographica," *Brains, Minds, and Media*, vol. 3, 2008. [Online]. Available: http://www.brains-minds-media.org/archive/1402

[75] ——, "Topographica: building and analyzing map-level simulations from Python, C/C++, MATLAB, NEST, or NEURON components," *Frontiers in Neuroinformatics*, vol. 3, 2009.

[76] M. Stimberg *et al.*, "Equation-oriented specification of neural models for simulations," *Frontiers in Neuroinformatics*, vol. 8, 2014.

[77] B. Ermentrout, *Simulating, analyzing, and animating dynamical systems: a guide to XPPAUT for researchers and students*. Siam, 2002, vol. 14.

[78] R. Clewley, "Hybrid models and biological model reduction with PyDSTool," *PLoS Comput Biol*, vol. 8, no. 8, p. e1002628, 2012.

[79] D. F. Goodman *et al.*, "Brian 2: neural simulations on a variety of computational hardware," *BMC Neuroscience*, vol. 15, no. Suppl 1, p. P199, 2014.

[80] M. Djurfeldt *et al.*, "Run-time interoperability between neuronal network simulators based on the MUSIC framework," *Neuroinformatics*, vol. 8, no. 1, pp. 43–60, 2010.

[81] B. Szigeti *et al.*, "Openworm: an open-science approach to modeling caenorhabditis elegans," *Frontiers in Computational Neuroscience*, vol. 8, 2014.

[82] T. Carnevale *et al.*, "The neuroscience gateway portal: high performance computing made easy," *BMC Neuroscience*, vol. 15, no. Suppl 1, p. P101, 2014.

[83] S. Sivagnanam *et al.*, "Introducing The Neuroscience Gateway." in *IWSG*. Citeseer, 2013.

[84] R. A. Poldrack and J.-B. Poline, "The publication and reproducibility challenges of shared data," *Trends in Cognitive Sciences*, vol. 19, no. 2, pp. 59–61, 2015.

[85] R. L. Henderson, "Job scheduling under the portable batch system," in *Job scheduling strategies for parallel processing*. Springer, 1995, pp. 279–294.

[86] A. B. Yoo *et al.*, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*. Springer, 2003, pp. 44–60.

[87] T. Yamazaki *et al.*, "Simulation platform: A cloud-based online simulation environment," *Neural Networks*, vol. 24, no. 7, pp. 693–698, 2011.

[88] S. Usui, "Visiome: neuroinformatics research in vision project," *Neural Networks*, vol. 16, no. 9, pp. 1293–1300, 2003.

[89] Y. O. Halchenko and M. Hanke, "Open is not enough. let's take the next step: an integrated, community-driven computing platform for neuroscience," *Frontiers in Neuroinformatics*, vol. 6, 2012.

[90] R. C. Cannon *et al.*, "LEMS: a language for expressing complex biological models in concise and hierarchical form and its use in underpinning NeuroML 2," *Frontiers in Neuroinformatics*, vol. 8, 2014.

[91] D. Beeman, "History of neural simulation software," in *20 Years of Computational Neuroscience*. Springer, 2013, pp. 33–71.

[92] A. P. Davison, "Collaborative modelling: The future of computational neuroscience?" *Network: Computation in Neural Systems*, vol. 23, no. 4, pp. 157–166, 2012.

[93] R. Brette *et al.*, "Simulation of networks of spiking neurons: a review of tools and strategies," *Journal of Computational Neuroscience*, vol. 23, no. 3, pp. 349–398, 2007.

[94] R. C. Gerkin and C. Omar, "Collaboratively testing the validity of neuroscientific models," *Frontiers in Neuroinformatics*, p. 1, 2014.

[95] R. C. Cannon *et al.*, "Interoperability of neuroscience modeling software: current status and future directions," *Neuroinformatics*, vol. 5, no. 2, pp. 127–138, 2007.

[96] R. Cannon *et al.*, "An on-line archive of reconstructed hippocampal neurons," *Journal of Neuroscience Methods*, vol. 84, no. 1, pp. 49–54, 1998.

[97] G. A. Ascoli *et al.*, "NeuroMorpho.Org: a central resource for neuronal morphologies," *The Journal of Neuroscience*, vol. 27, no. 35, pp. 9247–9251, 2007.

[98] T. Bray *et al.*, "Extensible markup language (XML)," *World Wide Web Consortium Recommendation REC-xml-19980210. http://www.w3. org/TR/1998/REC-xml-19980210*, vol. 16, 1998.

[99] M. Hucka *et al.*, "Promoting coordinated development of community-based information standards for modeling in biology: the COMBINE initiative," *Frontiers in Bioengineering and Biotechnology*, vol. 3, 2015.

[100] A. R. Ferguson *et al.*, "Big data from small data: data-sharing in the 'long tail' of neuroscience," *Nature Neuroscience*, vol. 17, no. 11, pp. 1442–1447, 2014.

[101] A. I. for Brain Science, "Allen cell types database [internet]," 2015. [Online]. Available: http://celltypes.brain-map.org

[102] C. M. Lloyd *et al.*, "CellML: its future, present and past," *Progress in Biophysics and Molecular Biology*, vol. 85, no. 2, pp. 433–450, 2004.

[103] H. Peng *et al.*, "BigNeuron: large-scale 3D neuron reconstruction from optical microscopy images," *Neuron*, vol. 87, no. 2, pp. 252–256, 2015.

[104] S. Wearne *et al.*, "New techniques for imaging, digitization and analysis of three-dimensional neural morphology on multiple scales," *Neuroscience*, vol. 136, no. 3, pp. 661–680, 2005.

[105] D. R. Myatt *et al.*, "Neuromantic–from semi-manual to semi-automatic reconstruction of neuron morphology," *Frontiers in Neuroinformatics*, vol. 6, 2012.

[106] M. Vella *et al.*, "libNeuroML and PyLEMS: using Python to combine procedural and declarative modeling approaches in computational neuroscience," *Frontiers in Neuroinformatics*, vol. 8, 2014.

[107] P. Gleeson *et al.*, "Using NeuroML and neuroConstruct to build neuronal network models for multiple simulators," *BMC Neuroscience*, vol. 8, no. Suppl 2, p. P1, 2007.

[108] I. Raikov *et al.*, "NineML: the network interchange for neuroscience modeling language," *BMC Neuroscience*, vol. 12, no. Suppl 1, p. P330, 2011.

[109] W. J. Hedley *et al.*, "A short introduction to CellML," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 359, no. 1783, pp. 1073–1089, 2001.

[110] A. A. Cuellar *et al.*, "An overview of CellML 1.1, a biological model description language," *Simulation*, vol. 79, no. 12, pp. 740–747, 2003.

[111] D. Waltemath *et al.*, "Reproducible computational biology experiments with SED-ML-the simulation experiment description markup language," *BMC Systems Biology*, vol. 5, no. 1, p. 198, 2011.

[112] F. T. Bergmann and H. M. Sauro, "Comparing simulation results of SBML capable simulators," *Bioinformatics*, vol. 24, no. 17, pp. 1963–1965, 2008.

[113] M. Mattioni and N. Le Novère, "Integration of biochemical and electrical signaling-multiscale model of the medium spiny neuron of the striatum," *PloS One*, vol. 8, no. 7, p. e66811, 2013.

[114] M. Schulz *et al.*, "SBMLmerge, a system for combining biochemical network models," *Genome Informatics*, vol. 17, no. 1, pp. 62–71, 2006.

[115] R. Ausbrooks *et al.*, "Mathematical markup language (MathML) version 2.0 . W3C recommendation," *World Wide Web Consortium*, vol. 2003, 2003.

[116] A. K. Miller *et al.*, "An overview of the CellML API and its implementation," *BMC Bioinformatics*, vol. 11, no. 1, p. 178, 2010.

[117] G. R. Christie *et al.*, "FieldML: concepts and implementation," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 367, no. 1895, pp. 1869–1884, 2009.

[118] R. D. Britten *et al.*, "FieldML, a proposed open standard for the Physiome project for mathematical model representation," *Medical & biological engineering & computing*, vol. 51, no. 11, pp. 1191–1207, 2013.

[119] ——, "FieldML," *Encyclopedia of Computational Neuroscience*, pp. 1182–1186, 2015.

[120] L. P. Smith *et al.*, "SBML and CellML translation in Antimony and JSim," *Bioinformatics*, vol. 30, no. 7, pp. 903–907, 2014.

[121] M. Courtot *et al.*, "Controlled vocabularies and semantics in systems biology," *Molecular Systems Biology*, vol. 7, no. 1, Oct. 2011. [Online]. Available: http://dx.doi.org/10.1038/msb.2011.77

[122] D. Taichman *et al.*, "Sharing clinical trial data: a proposal from the international committee of medical journal editors," *Lancet*, 2016.

[123] G. A. Ascoli, "The ups and downs of neuroscience shares," *Neuroinformatics*, vol. 4, no. 3, pp. 213–215, 2006.

[124] ——, "Sharing neuron data: Carrots, sticks, and digital records," *PLOS Biol*, vol. 13, no. 10, p. e1002275, 2015.

[125] S. Ramaswamy *et al.*, "The neocortical microcircuit collaboration portal: a resource for rat somatosensory cortex," *Frontiers in Neural Circuits*, vol. 9, 2015.

[126] K. J. Gorgolewski *et al.*, "Making data sharing count: a publication-based solution," *Frontiers in Neuroscience*, vol. 7, 2013.

[127] D. Gardner *et al.*, "Towards effective and rewarding data sharing," *Neuroinformatics*, vol. 1, no. 3, pp. 289–295, 2003.

[128] E. Nordlie *et al.*, "Towards reproducible descriptions of neuronal network models," *PLoS Comput. Biol*, vol. 5, no. 8, p. e1000456, 2009.

[129] E. De Schutter, "The dangers of plug-and-play simulation using shared models," *Neuroinformatics*, vol. 12, no. 2, p. 227, 2014.

[130] M. L. Hines *et al.*, "ModelDB: a database to support computational neuroscience," *Journal of Computational Neuroscience*, vol. 17, no. 1, pp. 7–11, 2004.

[131] P. Gleeson *et al.*, "The Open Source Brain Initiative: enabling collaborative modelling in computational neuroscience," *BMC Neuroscience*, vol. 13, no. Suppl 1, p. O7, 2012.

[132] M. Halavi *et al.*, "Digital reconstructions of neuronal morphology: three decades of research trends," *Frontiers in Neuroscience*, vol. 6, 2012.

[133] M. Kohn *et al.*, "A block organized model builder," *Mathematical and Computer Modelling*, vol. 19, no. 6, pp. 75–97, 1994.

[134] M. L. Hines and N. T. Carnevale, "Expanding NEURON's repertoire of mechanisms with NMODL," *Neural Computation*, vol. 12, no. 5, pp. 995–1007, 2000.

[135] C. M. Lloyd *et al.*, "The CellML model repository," *Bioinformatics*, vol. 24, no. 18, pp. 2122–2123, 2008.

[136] T. Yu *et al.*, "The physiome model repository 2," *Bioinformatics*, vol. 27, no. 5, pp. 743–744, 2011.

[137] N. Juty *et al.*, "BioModels: Content, Features, Functionality and Use." *CPT: Pharmacometrics and Systems Pharmacology*, 2015.

[138] D. Gardner *et al.*, "The Neuroscience Information Framework: a data and knowledge environment for neuroscience," *Neuroinformatics*, vol. 6, no. 3, pp. 149–160, 2008.

[139] A. Gupta *et al.*, "Federated access to heterogeneous information resources in the Neuroscience Information Framework (nif)," *Neuroinformatics*, vol. 6, no. 3, pp. 205–217, 2008.

[140] B. G. Olivier and J. L. Snoep, "Web-based kinetic modelling using JWS Online," *Bioinformatics*, vol. 20, no. 13, pp. 2143–2144, 2004.

[141] J. L. Teeters *et al.*, "Data sharing for computational neuroscience," *Neuroinformatics*, vol. 6, no. 1, pp. 47–55, 2008.

[142] S. Ray *et al.*, "NSDF: Neuroscience simulation data format," *Neuroinformatics*, pp. 1–21, 2015.

[143] J. L. Teeters *et al.*, "Neurodata Without Borders: Creating a common data format for neurophysiology," *Neuron*, vol. 88, no. 4, pp. 629–634, 2015.

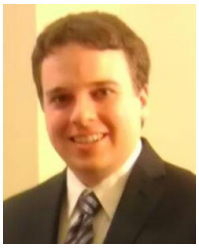
[144] M. Folk *et al.*, "HDF5: A file format and I/O library for high performance computing applications," in *Proceedings of Supercomputing*, vol. 99, 1999, pp. 5–33.

[145] I. Raikov and E. De Schutter, "The promise and shortcomings of XML as an interchange format for computational models of biology," *Neuroinformatics*, vol. 10, no. 1, pp. 1–3, 2012.

[146] N. Le Novere *et al.*, "BioModels database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems," *Nucleic Acids Research*, vol. 34, no. suppl 1, pp. D689–D691, 2006.

[147] J. Cooper *et al.*, "A call for virtual experiments: accelerating the scientific process," *Progress in Biophysics and Molecular Biology*, vol. 117, no. 1, pp. 99–106, 2015.

[148] L. Marenco *et al.*, "The NIF LinkOut broker: a web resource to facilitate federated data integration using NCBI identifiers," *Neuroinformatics*, vol. 6, no. 3, pp. 219–227, 2008.
[149] D. J. Hamilton *et al.*, "An ontological approach to describing neurons and their relationships," *Frontiers in Neuroinformatics*, vol. 6, 2012.
[150] S. D. Larson and M. E. Martone, "NeuroLex.org: an online framework for neuroscience knowledge," *Frontiers in Neuroinformatics*, vol. 7, 2013.
[151] A. Bandrowski *et al.*, "The Resource Identification Initiative: A cultural shift in publishing," *Journal of Comparative Neurology*, vol. 524, no. 1, pp. 8–22, 2016.
[152] Y. Le Franc *et al.*, "Computational Neuroscience Ontology: a new tool to provide semantic meaning to your models," *BMC Neuroscience*, vol. 13, no. Suppl 1, p. P149, 2012.
[153] D. Waltemath *et al.*, "Minimum information about a simulation experiment (MIASE)," *PLoS Computational Biology*, vol. 7, no. 4, pp. e1 001 122_1–e1 001 122_4, 2011.
[154] M. Bezaire and I. Soltesz, "Quantitative assessment of CA1 local circuits: Knowledge base for interneuron-pyramidal cell connectivity," *Hippocampus*, vol. 23, pp. 751–785, 2013.
[155] J. Antolík and A. P. Davison, "Integrated workflows for spiking neuronal network simulations," *Frontiers in Neuroinformatics*, vol. 7, 2013.
[156] J.-L. R. Stevens *et al.*, "An automated and reproducible workflow for running and analyzing neural simulations using Lancet and IPython Notebook," *Frontiers in Neuroinformatics*, vol. 7, 2013.
[157] A. Davison, "Automated capture of experiment context for easier reproducibility in computational research," *Computing in Science & Engineering*, vol. 14, no. 4, pp. 48–56, 2012.
[158] A. P. Davison *et al.*, "Sumatra: A toolkit for reproducible research," *Implementing Reproducible Research*, p. 57, 2014.
[159] D. Waltemath *et al.*, "Improving the reuse of computational models through version control," *Bioinformatics*, vol. 29, no. 6, pp. 742–748, 2013.
[160] D. W. Wheeler *et al.*, "Hippocampome.org: a knowledge base of neuron types in the rodent hippocampus," *Elife*, vol. 4, p. e09960, 2015.
[161] S. J. Tripathy *et al.*, "NeuroElectro: a window to the world's neuron electrophysiology data," *Frontiers in Neuroinformatics*, vol. 8, 2014.
[162] K. Popper, *Conjectures and refutations; the growth of scientific knowledge*. NY: Basic Books, 1962.



Robert A. McDougal received his Ph.D. in Mathematics from The Ohio State University in 2011 and a M.S. in Computational Biology and Bioinformatics from Yale University in 2015. He has over four years experience as a developer for the NEURON simulator and for the ModelDB repository, and he is currently a Postdoctoral Fellow in the Department of Neuroscience at Yale University researching simulator development and model analysis.

Anna S. Bulanova received a Diploma in Applied Mathematics at St. Petersburg State University, Russia in 2002; M.Sc. and Ph.D. in Mathematics from University of Alaska Fairbanks in 2006 and 2009 respectively. In 2012 she started a postdoctoral job at INRIA, France, to work in a project concerned with applying neuromorphic architectures to computing purposes. Since 2013 she is a Postdoctoral Researcher with NEURON simulator development group at Yale University and SUNY Downstate Medical Center. Her current research interests include Computational Neuroscience, Control Theory, and Signal Processing.

William W. Lytton is a Professor in Physiology, Pharmacology and Neurology at SUNY Downstate and works as a clinical neurologist at Kings County Hospital, seeing patients with a variety of brain ailments. His research is in Computational Neuroscience with a focus on the application of multiscale modeling to various disorders of the brain, including schizophrenia, dystonia, epilepsy, Alzheimers, and stroke. He is author of "From Computer to Brain" a basic introduction to the field.