SimpleSBML: A Python package for creating, editing, and interrogating SBML models: Verison 2.0

Herbert M. Sauro[*]

Department of Bioengineering,
University of Washington, Seattle, WA 98195, USA

September 2, 2020

Abstract

In this technical report, we describe a new version of SimpleSBML which provides an easier to use interface to python-libSBML allowing users of Python to more easily construct and inspect SBML based models. The most commonly used package for constructing SBML models in Python is python-libSBML based on the C/C++ library libSBML. python-libSBML is a comprehensive library with a large range of options but can be difficult for new users to learn and requires long scripts to create even the simplest models. Inspecting existing SBML models can also be difficult due to the complexity of the underlying object model. Instead, we present SimpleSBML, a package that allows users to add and inspect species, parameters, reactions, events, and rules to a libSBML model with only one command for each. Models can be exported to SBML format, and SBML files can be imported and converted to SimpleSBML commands that create each element in a new model. This allows users to create new models and edit existing models for use with other software. In the new version, a range of 'get' methods is provided that allows users to inspect existing SBML models without having to understand the underlying object model used by libSBML.

Accessibility and Implementation:

SimpleSBML is publicly available and licensed under the liberal MIT open source license. It supports SBML levels 2 and 3. Its only dependency is libSBML. It is supported on Windows and Mac OS X but should also work on Linux (untested). All code has been deposited at the GitHub site https://github.com/sys-bio/simplesbml and is available for user installation via a standard `pip install simplesbml`.

Contact:

hsauro@uw.edu

Supplementary information:

Documentation and download is available at sys-bio.github.io/simplesbml as well as a pip install on pypi

---

[*]to whom correspondence should be addressed

1   Introduction

In an earlier report [4] we described SimpleSBML, a python library that makes it much easier to create SBML models. In this document, we describe a new version of simpleSBML that extends the functionality of simpleSBML to now include a series of `get` methods to easily extract information from an existing SBML model.

Biological modeling is a key component of systems biology, and the development of Systems Biology Markup Language (SBML) [1], a markup language designed to describe models of biological systems, has allowed systems and synthetic biologists to develop a plethora of useful software tools that are automatically compatible with each other through SBML document import and export. Creating SBML models is possible with a variety of software packages, the most well-known of which is libSBML [2]. This package allows users to generate SBML documents by writing scripts in Python, C or C++. While libSBML is very useful, it can be difficult to learn for novices and even the simplest of models using libSBML tend to be very long. SimpleSBML is a package that also allows users to create SBML models with Python scripting, but requires far fewer commands and is more accessible for beginner users. One other difference between version 1 and 2 is that the class name has been capitalized to match common practice in Python code, hence use `SbmlModel` instead of `sbmlModel`.

SimpleSBML makes use of libSBML methods to add and isspect elements such as species, parameters, reaction, events and rules to a model, and generate an SBML-formatted document from the resulting model object. SimpleSBML also includes two new methods that allow users to load existing SBML modls either from a string or file. SimpleSBML can therefore be used to edit SBML-formatted models as well as create new ones.

2   Methods

Verison 1.0 of simplesbml is given in the previous publication [4] and the examples for creating a new SBML model will not be repeated other than giving one example for reference. The SbmlModel class, holds a SBMLDocument object and contains methods that can be used to add different elements, such as species, parameters, reactions and events. Here is an example of a simple reaction-based model built with SbmlModel.

```python
import simplesbml
model = simplesbml.sbmlModel();
model.addCompartment(1e-14, comp_id='comp');
model.addSpecies('E', 5e-21, comp='comp');
model.addSpecies('S', 1e-20, comp='comp');
model.addSpecies('P', 0.0, comp='comp');
model.addSpecies('ES', 0.0, comp='comp');
model.addReaction(['E', 'S'], ['ES'], 'comp*(kon*E*S-koff*ES)', \
        local_params={'koff': 0.2, 'kon': 1000000.0}, rxn_id='veq');
model.addReaction(['ES'], ['E', 'P'], 'comp*kcat*ES', \
                local_params={'kcat': 0.1}, rxn_id='vcat');
```

2.1 New get API

Of more interest is the new `get` API that allows an existing SBML model to be easily inspected. This section will discuss the new `get` API that verison 2.0 has. The first change is to allow existing SBML models to be loaded into simpleSBML. The existing constructor `SBMLModel` was modified to accept SBML strings and file names containing SBML. To make the interface simpler to use, two additional methods outside the class, `loadSBMLStr` and `loadSBMLFile` are provided. For example, to load a file one can use the following code:

```
model = simplesbml.loadSBMLFile ('mymodel.xml')
```

This returns an instance of `SbmlModel`. The online documentation gives a full list of methods that are available but a comment on the naming and class structure philosophy is in order. To begin with, the entire API is flat, that is there are no subclasses which one finds, for example, in libsbml. This means a user does not need to know the underlying object model in order to effectively use the API. However to make this work, the names for the API methods must be clear in what they do and can be identified using intellisense and code completion in IDEs (Integrated Development Environment) such as spyder (https://www.spyder-ide.org/). For example, to obtain the list of reaction Ids in a model requires the following code when using libsbml:

```
nReactions = document.model.getNumReactions ()
for i in range (nReactions):
    p = document.model.getReaction(i)
    print (p.getId())
```

where `document` was obtained by calling the libsbml method:

```
document = reader.readSBMLFromString(sbmlStr).
```

This requires three levels of indirection as well as knowledge of the various methods to call at each level. Instead, simplesbml allows a user to call a single method:

```
    print (model.getListOfReactionIds())
```

The method name is qualified with the term `Ids` because one could also retrieve the SBML names rather than the Ids. The entire API is built along these lines. As an illustration, the code below will use simplesbml to construct the stoichiometric matrix for a model. Tellurium [5] is used to convert the reaction scheme in antimony [6] format into SBML which is then loaded into simplesbml.

```
import tellurium as te, simplesbml,  numpy as np

r = te.loada("""
S0 + S3 -> S2; k0*S0*S3;
S3 + S2 -> S0; k1*S3*S2;
S5 -> S2 + S4; k2*S5;
S0 + S1 -> S3; k3*S0*S1;
S5 -> S0 + S4; k4*S5;
S0 -> S5; k5*S0;
S1 + S1 -> S5; k6*S1*S1;
S3 + S5 -> S1; k7*S3*S5;
S1 -> $S4 + S4; k8*S1;

S0 = 0; S1 = 0; S2 = 0; S3 = 0; S4 = 0; S5 = 0;
k0 = 0; k1 = 0; k2 = 0; k3 = 0; k4 = 0
```

```
    k5 = 0; k6 = 0; k7 = 0; k8 = 0
    """)

model = simplesbml.loadSBMLStr(r.getSBML())

stoich = np.zeros((model.getNumFloatingSpecies(), model.getNumReactions()))
for i in range (model.getNumFloatingSpecies()):
    floatingSpeciesId = model.getNthFloatingSpeciesId (i)

    for j in range (model.getNumReactions()):
        productStoichiometry = 0; reactantStoichiometry = 0

        numProducts = model.getNumProducts (j)
        for k1 in range (numProducts):
            productId = model.getProduct (j, k1)

            if (floatingSpeciesId == productId):
                productStoichiometry += model.getProductStoichiometry (j,k1)

        numReactants = model.getNumReactants(j)
        for k1 in range (numReactants):
            reactantId = model.getReactant (j, k1)
            if (floatingSpeciesId == reactantId):
                reactantStoichiometry += model.getReactantStoichiometry (j,k1)

        stoich[i,j] = int(productStoichiometry - reactantStoichiometry)
print (stoich)
```

The resulting stoichiometry matrix is given by:

```
[[-1.  1.  0. -1.  1. -1.  0.  0.  0.]
 [-1. -1.  0.  1.  0.  0.  0. -1.  0.]
 [ 1. -1.  1.  0.  0.  0.  0.  0.  0.]
 [ 0.  0. -1.  0. -1.  1.  1. -1.  0.]
 [ 0.  0.  0. -1.  0.  0. -2.  1. -1.]]
```

3   Installation

SimpleSBML depends on python-libSBML so that libSBML must be installed. Refer to the https://pypi.pyth
for details. SimpleSBML can installed using the standard pip mechanism for Python packages by
using the command:

```
pip install simplesbml
```

When using more sophisticated Python environments such as Jupyter (https://jupyter.org/),
Spyder or PyCharm (https://www.jetbrains.com/pycharm/), it is possible to type `pip install`
`simplesbml` at the Python console itself and the package will be installed automatically. If you're
using a more basic Python console such as Idle, you will need to use the pip command from
the operating system terminal. Make sure that Python and pip are on the search path if you

take this approach. On Windows the easiest way to get simplesbml is to install the Tellurium modeling package found at http://tellurium.analogmachine.org/. This comes ready bundled with simplesbml.

4 Testing and Documentation

Documentation is generated from comments in source code using sphinx and uploaded to readthe-docs. Additional documentation is provided by including text in the `index.rst` file.

Testing was originally done using the Python unittest (https://docs.python.org/3/library/unittest.html) package but was found to have limit capabilities. The missing functionality included an easy way to test the API on a variety of loaded SBML models and to exercise the API more thoroughly. Such a workflow is not supported well by unittest. Instead pyTest (https://docs.pytest.org/en/stable/) was tried which does support parameterized tests but the code became too verbose and unwieldy. Instead a very simple testing framework was written which was ultimately easier to add new tests to. The test file is called `runTests.py` and is located in the `tests` folder. To trun the tests, simple run this file from Python.

5 Discussion

SimpleSBML is intended for systems biology researchers who have limited experience with programming, or are working on simple models and prefer to use a simpler set of commands compared to libSBML. Future additions to the software may include additional methods to deal with model annotations.

Acknowledgments

References

[1] Hucka, M., A. Finney, H. M. Sauro, et al. (2003) The Systems Biology Markup Language (SBML): A Medium for Representation and Exchange of Biochemical Network Models. *Bioinformatics* 19(4): 524?531, doi:10.1093/bioinformatics/btg015.

[2] Bornstein, B.J., Keating, S. M., Jouraku, A., and Hucka M. (2008) LibSBML: An API library for SBML. *Bioinformatics*, 24(6):880-881, doi:10.1093/bioinformatics/btn051.

[3] Somogyi, E.T, Bouteiller, J-M, Glazier, J.A., König, M, Medley, K, Swat, M.H, Sauro, H.M. (2015) libRoadRunner: A High Performance SBML Simulation and Analysis Library. Bioinformatics. 2015 Oct 15;31(20):3315-21. https://doi.org/10.1093/bioinformatics/btv363

[4] Cannistra, Caroline, Medley, Kyle, Sauro, Herbert M., (2015), SimpleSBML: A Python package for creating and editing SBML models, doi = 10.1101/030312, bioRxiv

[5] Choi, K., Medley, J. K., Knig, M., Stocking, K., Smith, L., Gu, S., and Sauro, H. M. (2018). Tellurium: An extensible python-based modeling environment for systems and synthetic biology. Bio Systems, 171, 7479. https://doi.org/10.1016/j.biosystems.2018.07.006

[6] Smith, L. P., Bergmann, F. T., Chandran, D., and Sauro, H. M. (2009). Antimony: a modular model definition language. Bioinformatics (Oxford, England), 25(18), 24522454. https://doi.org/10.1093/bioinformatics/btp401