LIBSBML_DRAW: A PYTHON PACKAGE FOR BRINGING RENDER AND LAYOUT TO LIBSBML

A PREPRINT

Natalie Hawkins
Department of Bioengineering
University of Washington
Seattle, WA 98195
nhawkins@uw.edu

Kyle Medley
Department of Bioengineering
University of Washington
Seattle, WA 98195
medjk@comcast.net

Herbert Sauro
Department of Bioengineering
University of Washington
Seattle, WA 98195
hsauro@uw.edu

September 19, 2019

ABSTRACT

Summary: The SBML layout and render extensions enable SBML models to encode information about the graphical depiction of model elements. Layout provides information about the positions of model elements and render describes the styles of elements, for example, shapes, colors, line widths, and font details. In this application note, we describe `libsbml_draw` a Python package that supports the SBML layout and render extensions and can automatically generate a layout for SBML models by making use of SBNW, a C/C++ library.
Platforms: Windows, Linux, macOS
Software Availability:
Python package on the PyPI server: `pip install libsbml-draw`
SBNW C/C++ library on GitHub: `https://github.com/sys-bio/libsbml-draw`
Documentation: `https://libsbml-draw.readthedocs.io/en/latest/index.html`

***K*eywords** SBML · Python · libSBML

1 Introduction

SBML (Hucka et al., 2003) is the standard used for describing and sharing biochemical network models (Sauro, 2014)(Hoksza, 2019). A number of extensions have been made to SBML (Dräger et al., 2014), including the layout (Gauges et al., 2006) and render extensions (Gauges, 2009). The layout extension allows the size and positions of elements in a biochemical network to be specified. These elements can be species, compartments, reactions, and modifiers. The render extension allows for styles of the elements to be described. A style can specify the shape of a species or compartment, the color of an element or text, the width of a reaction or modifier line, a variety of font specifications for text, and line endings to use for reactions and modifiers.

Many SBML models have been created which do not contain layout information (Bergman and Sauro, 2006), which can be a barrier to interacting with the model via a graphical software tool. Thus, it is desirable to have a library which can compute and assign a layout to an SBML model.

In this application note, we present a Python package which allows for SBML models to be loaded, modified and written out again. The reading and writing of the SBML models is accomplished by using features of the libSBML library (Bornstein *et al.*, 2008). The models can be graphically displayed, and the image saved, which utilizes Python's matplotlib. If a loaded model does not contain layout information or if the user would like to override an existing layout, a layout can be auto-computed. The package utilizes an SBNW library written in C/C++ for auto-computing layouts. SBNW is based on the original SBW layout engine (Deckard et al., 2006), which was written in `C#`.

2 Package Features

2.1 Layout and Render Extension Support

The package can read and write SBML (Hucka et al., 2003) models which have layout and render information. If a model is lacking layout information, a layout can be auto-generated. If a model is lacking render information, the style information can be added. The Python API allows color, line width, and a variety of font style changes.

2.2 Autolayout Algorithm

The library can auto-compute layout information consisting of centroid coordinates for compartments, nodes, reactions, and modifiers for SBML models when a model is lacking layout data or when it is desired to override an existing layout. A modified version of the Fruchterman-Reingold (FR) algorithm is used when auto-generating a layout. The FR algorithm has shown to reproduce underlying symmetry and to be robust in the face of variegated graph topologies (Fruchterman and Reingold, 1991).

2.3 Fine-tuning layouts via Aliasing and Locking of Nodes

A layout can be auto-generated repeatedly until a most-desirable one is found. An auto-generated layout can be fine-tuned by aliasing and/or locking nodes in the layout.

2.3.1 Aliasing

Aliasing can be used to reduce congestion in a layout, where a lot of reactions are connected to a node. Aliasing a node means that a node of degree n can be decomposed into n alias nodes which are drawn separately. The layout can be regenerated after these alias nodes are created, in which case all the reactions will no longer converge into one node, but the separate n nodes can move around and the reactions will follow them. The FR-algorithm is particularly good at laying out these types of graphs without having overlapping edges (Fruchterman and Reingold, 1991).

2.3.2 Locking

Another way to fine-tune a layout is to use locking. One or more nodes can be locked before regenerating a layout. The positions of these nodes will not change.

2.4 Modify Render Information

Using the Python API color changes can be made to compartments, species, reaction and modifier curves, and species text. Edge colors and fill colors can be changed for shapes. Line widths can be changed for edges of shapes and for reaction and modifier curves. A variety of font properties can be changed for species text.

2.5 Line Endings Graphical Information

If line endings are supplied in the render information, these will be drawn. If they are missing, a default set of line endings will be used, based on the role of the curve. Roles that have been identified are: substrate, sidesubstrate, product, sideproduct, modifier, activator, and inhibitor.

2.6 Export Rendered Model

Via the Python API, an SBML model can be drawn and saved in either the PDF or PNG format. Python's matplotlib is used for the rendering.

2.7 Language Bindings

`libsbml_draw` provides a class SBMLlayout, which contains all the methods that can be used to load, write, modify, draw, and save images of the SBML models. This Python package relies on the SBNW library of C/C++ code. Both the Python package and the underlying SBNW C/C++ code are publicly available.

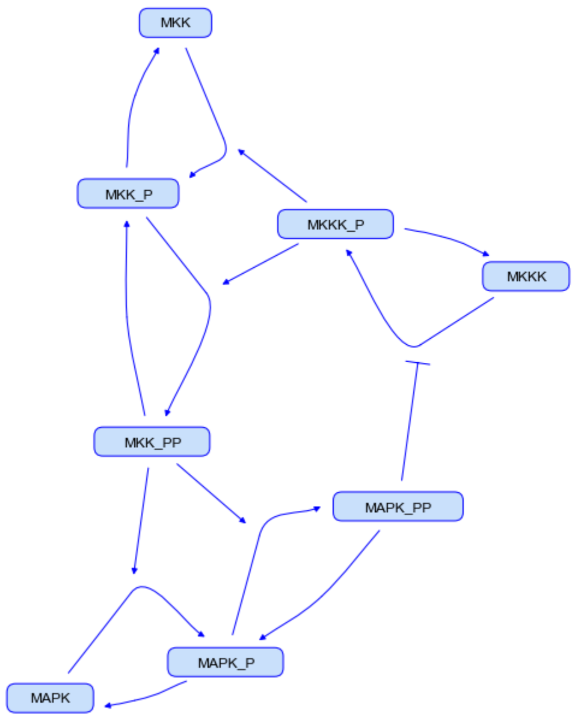

Figure 1: Boris EJB Model (Kholodenko, 2000)

3 Visualizations

We present two models loaded into and drawn by `libsbml_draw`.

In Figure 1, the model contained layout information, but we chose to auto-generate a layout instead. We also changed render styles.

In Figure 2, the model contained layout and render information, which we display.

4 Code Examples

```
from libsbml_draw import SBMLlayout
s = SBMLlayout("original_sbml_model.xml")
s.drawNetwork("my_sbml_model.pdf")
s.writeSBML("my_sbml_model.xml")
```

5 Summary

Installation of Python package: `pip install libsbml-draw`

Documentation: `https://libsbml-draw.readthedocs.io/en/latest/index.html`

SBNW C/C++ library: `https://github.com/sys-bio/libsbml-draw`

6 Future Work / Parts of the Render Standard Not Supported

There are some features of the render extension that `libsbml_draw` does not support. For example, linear gradients as fill colors are not supported, since matplotlib does not natively support linear gradients. Only 2D positional information

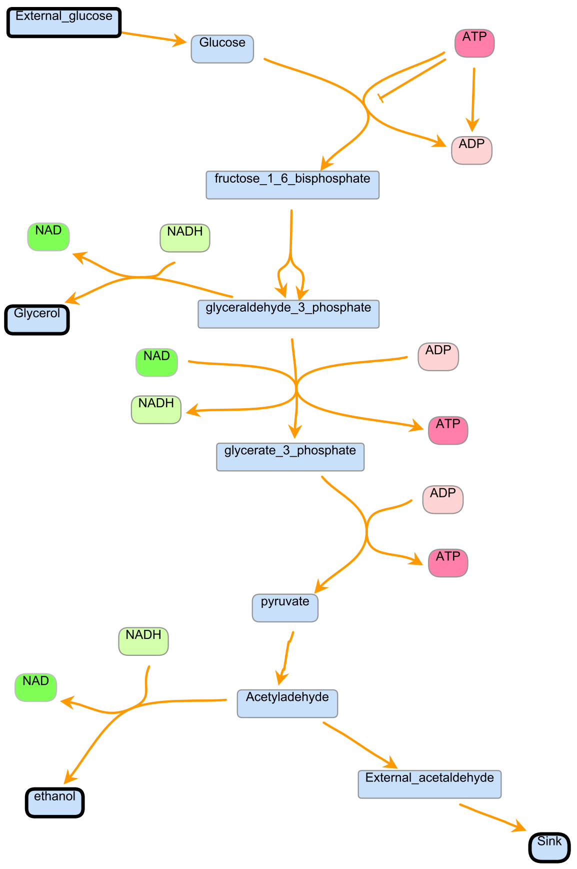

Figure 2: Glycolysis Model (Wolf and Heinrich, 2000)

is supported. Transformations are not supported, nor are styles which reference other styles. Also, if aliases are created, the output model file can only be read back into `libsbml_draw`.

Acknowledgments

We acknowledge Frank Bergmann for his help working with libsbml Python, and Joshua Katz for testing `libsbml_draw`.

Funding: The authors are most grateful to generous funding from the National Institute of General Medical Sciences of the National Institutes of Health under award P41-NIH. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

References

David Hoksza, Piotr Gawron, Marek Ostaszewski, Jan Hausenauer, and Reinhard Schneider. Closing the gap between formats for storing layout information in systems biology. *Briefings in Bioinformatics*, 00(00), 2019, 1-12.

Fruchterman, T.M.J. and Reingold, E.M. (1991) Graph drawing by force-directed placement, *Software: Practice and Ex perience*, **21** 11, 1129–1164.

Bergmann, F. T., and Sauro, H. M. (2006) SBW-a modular framework for systems biology, *Proceedings of the 38th conference on Winter simulation*, 1637–1645.

Bornstein, B, Keating, S, Jouraku, A, and Hucka (2008) LibSBML: an API Library for SBML, *Bioinformatics*, **24** 6, 880–881.

Deckard, A, Bergmann, F.T and Sauro H.M. (2006) Supporting the SBML layout extension, *Bioinformatics* **22** 23 2966–2967.

Dräger, A and Palsson B (2014) Improving Collaboration by Standardization Efforts in Systems Biology, *Frontiers in Bioengineering and Biotechnology* **2** 61.

Gauges, R, *et al* (2006) A model diagram layout extension for SBML, *Bioinformatics* **22** 15 1879-1885.

Gauges, R (2009) Complementing layout information with render information in SBML files. University of Heidelberg. Available at http://otto.bioquant.uni-heidelberg.de/sbml/level2/20091029/SBMLRenderExtension-20091029.pdf

Hucka, M, Finney, A, Sauro, H.M, Bolouri, H, Doyle, J, Kitano, H and the rest of the SBML Forum. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models.*Bioinformatics*, **19**(4), 524–531.

Kholodenko, B. N. (2000) Negative feedback and ultrasensitivity can bring about oscillations in the mitogen-activated protein kinase cascades, *European journal of biochemistry / FEBS*, **267** 6, 1583–1588.

Sauro H. M. (2014) Systems Biology: An Introduction to Pathway Modeling, Ambrosius Publishing, ISBN-10: 0982477376

Wolf, J and Reinhart H (2000) Effect of cellular interaction on glycolytic oscillations in yeast: a theoretical investigation, *Biochemical Journal*, **345**, 321–334.