Xml2jupyter: Mapping parameters between XML and Jupyter widgets

Randy Heiland[1], **Daniel Mishler**[1], **Tyler Zhang**[1], **Eric Bower**[1], and **Paul Macklin**[1]

1 Intelligent Systems Engineering, Indiana University

Summary

Jupyter Notebooks (Kluyver et al. 2016) provide executable documents (in a variety of programming languages) that can be run in a web browser. When a notebook contains graphical widgets, it becomes an easy-to-use graphical user interface (GUI). Many scientific simulation packages use text-based configuration files (hopefully in some standard format) to provide parameter values. Xml2jupyter is a Python package that bridges this gap. It provides a mapping between configuration files, formatted in the Extensible Markup Language (XML), and Jupyter widgets. Widgets are automatically generated from the XML file and these can, optionally, be incorporated into a larger GUI for a simulation package. Users modify parameter values via the widgets and the values are written to the XML configuration file. Xml2jupyter has been tested using the PhysiCell (Ghaffarizadeh et al. 2018) simulation software and will be used by students for classroom and research projects.

A PhysiCell configuration file defines model-specific user parameters in XML. Each parameter element consists of its name with attributes, defining its data *type* and *units* (optional), and the parameter's default value. The attributes will determine the appearance and behavior of the Jupyter widget. For numeric widgets (the most common type for PhysiCell), xml2jupyter will calculate a delta step size as a function of the default value and this step size will be used by the widget.

To illustrate, we show the following contrived XML example, containing each of the four allowed data types (currently):

```xml
<PhysiCell_settings>
  <user_parameters>
    <radius type="double" units="micron">250.0</radius>
    <threads type="int">8</threads>
    <color type="string">red</color>
    <debug type="bool">True</debug>
  </user_parameters>
</PhysiCell_settings>
```

When we map this into Jupyter widgets, we obtain the following rendered result (left). The middle snapshot shows the updated numeric value after the user does a single delta step (up) on the `radius` (note the step size of 10) and the right snapshot after the user single steps the `threads` value (note the step size of 1).

In another example, we use an XML configuration file from the *biorobots* sample project included with PhysiCell:

```xml
<user_parameters>
  <random_seed type="int" units="dimensionless">0</random_seed>

  <!-- for microenvironment setup -->
  <cargo_signal_D type="double" units="micron/min^2">1e3</cargo_signal_D>
  <cargo_signal_decay type="double" units="1/min">.4</cargo_signal_decay>
  <director_signal_D type="double" units="micron/min^2">1e3</director_signal_D>
  <director_signal_decay type="double" units="1/min">.1</director_signal_decay>

  <!-- for cell definitions -->
  <elastic_coefficient type="double" units="1/min">0.05</elastic_coefficient>
  ...
  <director_color type="string" units="none">limegreen</director_color>
</user_parameters>
```

To demonstrate this project, one would: 1) clone the repository (or download a release), 2) copy the sample XML configuration file to the root directory, 3) run the `xml2jupyter.py` script, providing the XML file as a argument.

```
$ cp config_samples/config_biorobots.xml .
$ python xml2jupyter.py config_biorobots.xml
```

The `xml2jupyter.py` script parses the XML and generates a Python module, `user_params.py`, containing the Jupyter widgets, together with methods to populate their values from the XML and write their values back to the XML. To "validate" the widgets were generated correctly, one could, minimally, open `user_params.py` in an editor and inspect it.

But to actually see the widgets rendered in a notebook, we provide a simple test:

```
$ python xml2jupyter.py config_biorobots.xml test_user_params.py
$ jupyter notebook test_gui.ipynb
```

This should produce the following notebook in your browser after selecting `Run all` in the `Cell` menu:
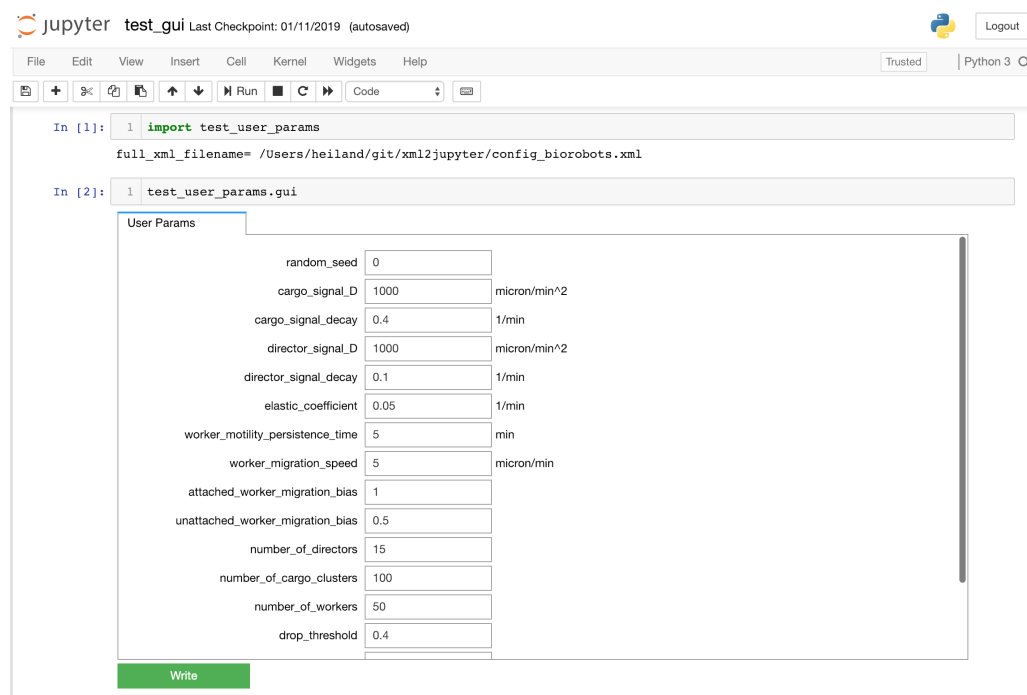
PhysiCell Jupyter GUI

Our ultimate goal is to generate a fully functional GUI for PhysiCell users. Xml2jupyter provides one important piece of this - dynamically generating widgets for custom user parameters for a model. With other Python modules that provide additional components (tabs) of the GUI, common to all PhysiCell models, a user can configure an entire simulation and visualize its output (we currently assume the user will run the simulation from a shell window). Two tabs that provide visualization of output files are shown below with results from the *biorobots* simulation. Note that some of the required modules are not available in the Python standard library, e.g., Matplotlib (Hunter 2007) and SciPy. We provide instructions for installing these additional dependencies in the repository README.
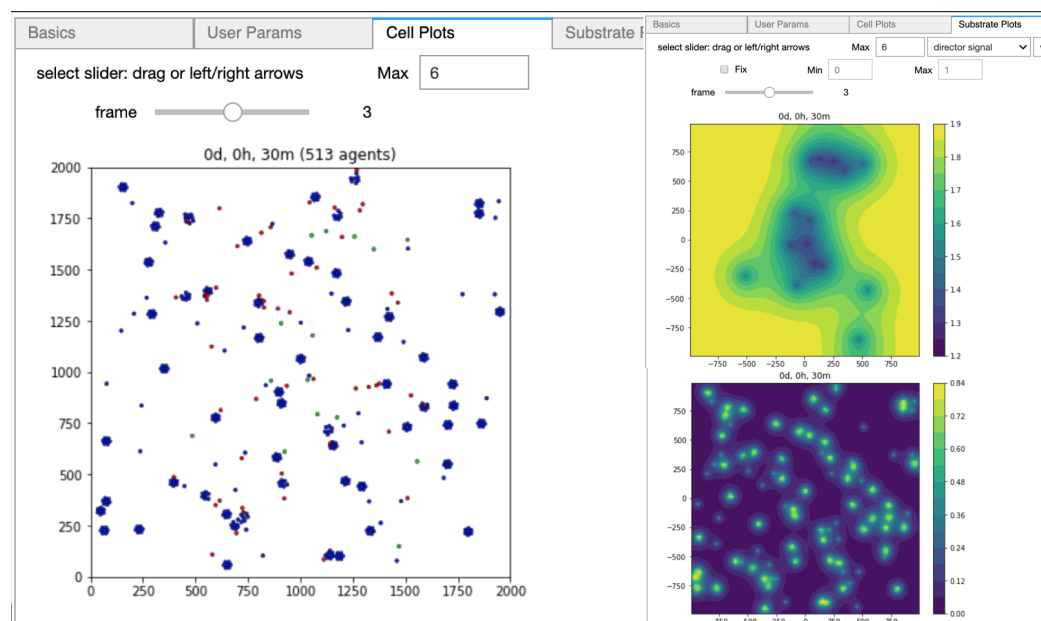
Extensions and Discussion

We hope others will be inspired to extend the core idea of this project to other text-based configuration files. XML is only one of several data-interchange formats. It just happens to be the one of interest to us for PhysiCell.

While the additional Python modules that provide visualization are specifically tailored to PhysiCell output, they can serve as templates for other file formats and provide similar functionality. We welcome contributions, especially for three-dimensional data.

Xml2jupyter has helped us port PhysiCell-related Jupyter tools to nanoHUB (Madhavan et al. 2013), a scientific cloud for nanoscience education and research that includes running interactive or batch simulations. For example, we show screen shots from our `pc4cancerbots` tool running on nanoHUB, where the *User Params* tab has been generated using the `xml2jupyter.py` script. Readers can create an account on nanoHUB and run `pc4cancerbots` for themselves.
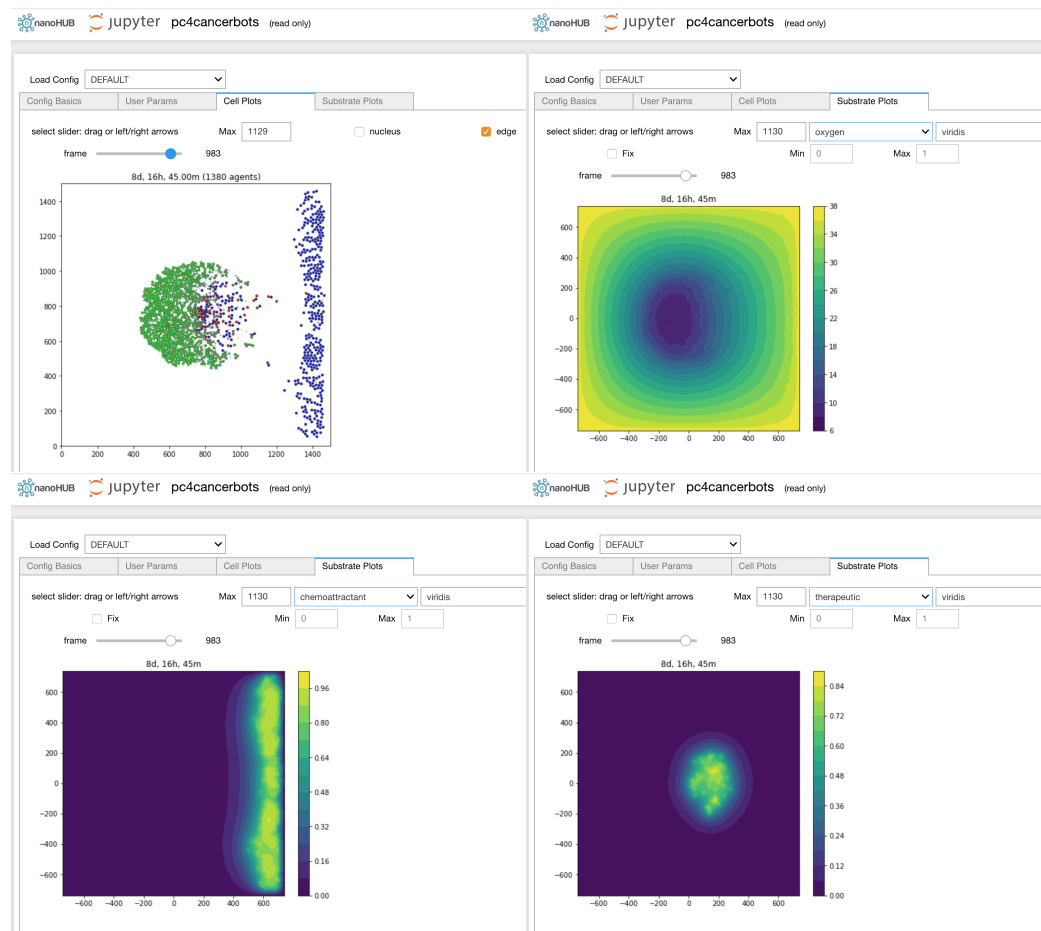
In the four screenshots below, we show the cell plot tab (upper-left) and three different substrate plots. This particular model and simulation is described in this video.

We welcome suggestions and, better yet, contributions (via GitHub pull requests) to xml2jupyter. For example, currently, we simply arrange the generated widgets in a single column. It may be preferable to allow a matrix-style layout, using additional attributes in the XML. Also, it may be possible to provide more informative *tooltip* (balloon) text on a widget and, if so, could be provided by another XML attribute. Additional ideas for visualization are also welcome.

Acknowledgements

References

Ghaffarizadeh, Ahmadreza, Randy Heiland, Samuel H. Friedman, Shannon M. Mumenthaler, and Paul Macklin. 2018. "PhysiCell: An Open Source Physics-Based Cell Simulator for 3-d Multicellular Systems." *PLOS Computational Biology* 14 (2). Public Library of Science: 1–31. doi:10.1371/journal.pcbi.1005991.

Hunter, J. D. 2007. "Matplotlib: A 2d Graphics Environment." *Computing in Science & Engineering* 9 (3). IEEE COMPUTER SOC: 90–95. doi:10.1109/MCSE.2007.55.

Kluyver, Thomas, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, et al. 2016. "Jupyter Notebooks – a Publishing Format for Reproducible Computational Workflows." Edited by F. Loizides and B. Schmidt. IOS Press.

Madhavan, Krishna, Lynn Zentner, Victoria Farnsworth, Swaroop Shivarajapura, Michael Zentner, Nathan Denny, and Gerhard Klimeck. 2013. "NanoHUB.org: Cloud-Based Services for Nanoscale Modeling, Simulation, and Education." *Nanotechnology Reviews* 2 (1). doi:10.1515/ntrev-2012-0043.