WholeCellViz: Data visualization for whole-cell models

Ruby Lee[1,†], Jonathan R. Karr[2,†] & Markus W. Covert[1*]

[1]Department of Bioengineering, Stanford University, and [2]Graduate Program in Biophysics, Stanford University, 318 Campus Drive West, Stanford CA 94305, USA.

[†]These authors contributed equally to this work.
[*]To whom correspondence should be addressed. Email: mcovert@stanford.edu.

WholeCellViz is a web-based software program for visually analyzing whole-cell simulations. WholeCellViz enables visual analysis of several all aspects of cell physiology including:

- The cell mass, volume, and shape,
- The copy number of every metabolite, RNA, and protein,
- The flux of every metabolic reaction,
- The status of every molecular machine – DNA polymerase, RNA polymerase, ribosome, FtsZ ring, and
- The copy number, superhelicity, integrity, and DNA binding status of every chromosome.

Please see the tutorial and the about sections at wholecellviz.stanford.edu for more information about WholeCellViz including how to use WholeCellViz and how it was implemented.

This document provides instructions on how to install, develop, and use WholeCellViz. Please contact the authors with any questions; updated contact information is available at wholecellviz.stanford.edu.

WholeCellViz is freely available at wholecellviz.stanford.edu. The WholeCellViz source code is available open-source at `http://simtk.org/home/wholecell`.

Contents

Chapter 1

Installation

1. Download the source code from SimTK (`http://simtk.org/home/wholecell`) and extract to `/path/to/WholeCellViz/`
2. Edit WholeCellViz configuration (`/path/to/WholeCellViz/configuration.php`)
   (a) Edit host, database, user, and password for your MySQL database
   (b) Edit location of simulated data (`simulationsBaseDir`)
   (c) Edit location of WholeCellViz (`WholeCellVizDataPath`, `WholeCellVizURL`, `WholeCellVizDataURL`)
3. Install Apache, MySQL, PHP, phpMyAdmin (eg. `http://www.wampserver.com/`)
4. Setup Apache
   (a) Add alias and/or virtual host
5. Setup MySQL
   (a) Set root password

   ```
   UPDATE mysql.user SET Password=PASSWORD('<rootpass>') WHERE User='root';
   FLUSH PRIVILEGES;
   ```

   (b) Remove test database
   (c) Remove anonymous users
   (d) Create database

   ```
   mysql >> create database <database>
   ```

   (e) Create user and grant permissions

   ```
   mysql >> CREATE USER '<user>'@'%' IDENTIFIED BY '<password>';
   mysql >> GRANT ALL ON <database>.* TO '<user>'@'%' WITH GRANT OPTION;
   ```

   (f) Load tables and data

   ```
   >> mysql -h <host> -u <user> --password=<password> <database> < \
   /path/to/wholecellviz/data/data.sql
   ```

6. Setup PHP
   (a) Add PHP to system path
   (b) Edit php.ini

   ```
   error_reporting = E_ALL & ~E_NOTICE
   display_errors = Off
   ```

7. Edit phpMyAdmin configuration

   ```
   $cfg['Servers'][$i]['auth_type'] = 'cookie';
   ```

8. Export simulation data to JSON format
   - Use the `runExportData` script from the WholeCell package to export each simulation batch

   ```
   /path/to/WholeCell/simulation/runExportData.pl <batch> <numSimulations>
   ```

- This will create files with the pattern `/path/to/WholeCell/simulation/output/runSimulation/<batch>/<sim>/json/<class_name>/<attr_name>.json`
- Note: this currently only exports a subset of the available data. Edit /path/to/WholeCell/simulation/runExportData.m to configure which states are exported.

9. Add simulations to `simulations` table in WholeCellKB database

```
>> php indexSimulations.php
```

Chapter 2

Developing visualizations

WholeCellViz was developed primarily in JavaScript using several libraries including jQuery (`http://jquery.com/`) and flot (`http://www.flotcharts.org/`). Additionally, several server-side web services where implemented in PHP.

There are two steps to create a new visualization. First, add a new JavaScript class to `/path/to/wholecellviz/js/WholeCellViz-visualizations.js` which extends the `Visualization` class defined in `/path/to/whole\-cell\-viz/\-js/\-Whole\-Cell\-Viz.js`. Visualizations should implement four methods:

- *`getData`* – This method should fetch data from external sources based on the selected simulation (`metadata.sim_id`). This method can call the `getSeriesData` helper method to fetch JSON-formatted data. This method should also set the visualization's `timeMin` and `timeMax` properties to inform the time range displayed to the user. This method is called once during the visualization's construction.
- *`calcLayout`* – This method should calculate the visualization's layout. This method is called during the visualization's construction and each time it is resized.
- *`drawStaticObjects`* – This method should draw parts of the visualization which don't change with time. This method is called during the visualization's construction and each time it is resized.
- *`drawDynamicObjects`* – This method should draw parts of the visualization which change over time. The `getDataPoint` and `getDataIndex` methods can be used to retrieve data based on the current time. This method is called during the visualization's construction, each time it is resized, and each time the animation time line is advanced.

Both the `drawStaticObjects` and `drawDynamicObjects` methods should use the `drawObject` method to draw objects to the canvas and to bind tool tips and mouse click handlers to the visualization.

Box 2.1 provides a template for constructing new visualizations. `/path/to/whole\-cell\-viz/js/Whole\-Cell\-Viz-vis\-ualizations.js` provides several sample visualizations.

Box 2.1 | Visualization class template.

```
var NewVisualization = Visualization.extend({
        getData: function(metadata){
                this.data = this.getSeriesData('getSeriesData.php', {
                        sim_id:metadata.sim_id,
                        class_name: '<class-name>',
                        attr_name: '<attr-name>',
                });
                this.timeMin = yourFuncToCalcMinTime(this.data);
                this.timeMax = yourFuncToCalcMaxTime(this.data);
        },
```

```
        calcLayout: function(){
        },

        drawStaticObjects: function(){
                this.drawObject({
                        'strokeStyle': strokeStyle,
                        'fillStyle': fillStyle,
                        'data': getDataPoint(this.data, t),
                        drawFunc: function(self, ctx, data){
                        },
                        tipFunc: function(self, data){
                        },
                        clickFunc: function(self, data){
                        },
                });
        },

        drawDynamicObjects: function(t){
                this.drawObject({
                        'strokeStyle': strokeStyle,
                        'fillStyle': fillStyle,
                        'data': getDataPoint(this.data, t),
                        drawFunc: function(self, ctx, data){
                        },
                        tipFunc: function(self, data){
                        },
                        clickFunc: function(self, data){
                        },
                });
        },
});
```

Second, add an entry for the visualization to the `attributes` table of the WholeCellKB database, using phpMyAdmin for example. Finally, navigate to WholeCellViz and use the configuration window to display your new visualization.

Chapter 3

Using WholeCellViz

To use WholCellViz first navigate to `http://path/to/your/server/WholeCellViz`. Next, to configure the panels either click the configuration button at the top-right of each panel or access the configuration window from the Edit button in the menu at the top-left of the page. Use the configuration window to select the simulation and data series plotted in each panel.

See the online tutorial for additional help using WholeCellViz.

Appendix A

License