Abstract

Background: At the molecular level, nonlinear networks of heterogeneous molecules control many biological processes, so that systems biology provides a valuable approach in this field, building on the integration of experimental biology with mathematical modeling. One of the biggest challenges to making this integration a reality is that many life scientists do not possess the mathematical expertise needed to build and manipulate mathematical models well enough to use them as tools for hypothesis generation. Available modeling software often assume some modeling expertise. There is a need for software tools that are easy to use and intuitive for experimentalists.

Results: This paper introduces PlantSimLab, a web-based application developed to allow plant biologists to construct dynamic mathematical models of molecular networks, interrogate them in a manner similar to what is done in the laboratory, and use them as a tool for biological hypothesis generation. It is designed to be used by experimentalists, without direct assistance from mathematical modelers.

Conclusions: Mathematical modeling techniques are a useful tool for analyzing complex biological systems and there is a need for accessible efficient analysis tools in the biological community. PlantSimLab enables users to build, validate, and use intuitive qualitative dynamic computer models, with a graphical user interface that does not require mathematical modeling expertise. It makes analysis of complex models accessible to a larger community, as it is platform-independent and does not require extensive mathematical expertise.

Keywords: mathematical model, biological network, dynamic network model, plant biology, modeling software.

BACKGROUND

Motivation

“Like most mathematicians, he takes the hopeful biologist to the edge of a pond, points out that a good swim will help his work, and then pushes him in and leaves him to drown.” (C. Elton, in a 1935 review of work by A. Lotka) [1]. The modern biologist might well have the same reaction when confronted with many of today’s mathematical models and software tools to build and use them. The ideal approach to (plant) systems biology that avoids this problem might be interdisciplinary research teams integrating biology and computation, with significant overlap in skill sets. This is unfortunately not the every-day reality in the short or medium term. A widespread adoption of tools that bring computation and systems “thinking” to the study of dynamic molecular pathways requires that a biologist can use them without access to a modeler and without having to acquire advanced mathematical skills. PlantSimLab, the software package described in this paper represents an attempt to address this reality. We aimed to build an intuitive
tool with a shallow learning curve and some basic utilities, a “Chevy” of modeling tools for the uninitiated, rather than a “Cadillac.”

Systems biology is a useful approach to plant biology, and biology in general, at several different scales (see, e.g., [2]). Focusing on system-level dynamic phenomena naturally relies on extensive use of mathematical models. If the interest is in elucidating network topology, then typically tools from graph theory are used to study connectivity features. If the interest is in studying dynamic effects, then one needs to build and analyze dynamic computer models of networks. In both cases, two approaches are used: the so-called “bottom-up” approach, which builds a model of the network from available information about its components, and the “top-down” approach, which extracts network links from experimental, typically high-throughput, data. Ideally, both approaches are used in combination. The software tool described here in its current form is entirely bottom-up. It provides a graphical user interface that allows the construction of dynamic models of networks, their simulation, and the basic experimental capability of knocking out a node. The fundamental hypothesis underlying PlantSimLab is that with the right type of modeling paradigm and the right interface, biologists can themselves construct and manipulate useful mathematical models for hypothesis generation, without having expert knowledge or background in mathematics or modeling. The goal is to provide biologists with a virtual laboratory tool to integrate available information and data for the purpose of hypothesis generation, easily usable, with a shallow learning curve. These constraints require a mathematically simple way of constructing models and of being able to interpret model output, making modeling frameworks like differential equations less well-suited. We have chosen the modeling paradigm of time- and state-discrete dynamical systems, essentially generalized Boolean networks. They have been used successfully to capture a wide range of molecular networks in recent years; see, e.g., [3-6].

**Existing software packages**

There are a number of excellent software platforms available for modeling using Boolean networks and their generalizations. The Cell Collective [7] is an interactive web tool built with a special emphasis on the collaboration of distributed teams to build large Boolean models of molecular networks, that has many features and an extensive library of models. The popular web tool GinSim [8] provides a graphical user interface to build and analyze so-called logical models, which can be viewed as generalized Boolean networks with additional features. The R package BoolNet [9] is a very convenient approach to building and simulating stochastic Boolean models with randomly varying update schemes for the variables. Several other more recent platforms have been developed; see, e.g., [6, 10-13] for an incomplete list. Some of these are equipped to accommodate only Boolean networks, such as BoolNet, others can handle multi-state models, such as GinSim. They have in common, to a greater or lesser extent, the assumption that the user has some modeling experience or is willing to admit to a relatively steep learning curve. And they all have a wide variety of features, whether it is the accommodation of different modeling frameworks, the ability to infer dynamic models from data, or features that allow sharing and distributed model construction.
Our contribution
The modeling platform described here differs from these platforms primarily through its greater simplicity, relative lack of features, and ubiquitous default settings that shorten the path to a working model as much as possible. The user can choose any (finite) number of states for any of the nodes, with state labels chosen from a predetermined menu (with the option to customize). The user then chooses edges between nodes, from a set of default choices (e.g., activate, inhibit, custom). For activating or inhibiting edges, an autofilled transition table appears (that can be customized, if desired) that specifies the action of the edge, taking into account the respective numbers of states for source and target nodes and their labels. We have chosen to use basic transition tables to describe the logical rules for the way each node takes up and integrates its different regulatory inputs. The advantage is that, in essence, each row in a transition table represents a biological statement, such as “when A is high, B is low, and C is low at time t, then C (which is regulated by both) transitions to being medium at time t+1,” corresponding to the row [high low low | medium], which, for mathematical purposes gets translated into [2 0 0 | 1]. Thus, model construction is simplified as much as possible. Model analysis in the current version is limited essentially to computing the attractors of the system and the basins of attraction. Basic “experiments” that can be performed with the model include “knock-out” of one or more nodes and the edges connected to the knocked-out node(s), and the ability to observe the resulting changes in dynamics. In our experience, this “bare-bones” approach is effective in providing a quick way to model construction and a check on the consistency of the assumptions underlying the model. This is the first step in using the model for hypothesis discovery. The existing modeling platforms described above then serve as an “upgrade” for the biologist from our basic framework.

IMPLEMENTATION
The purpose of the software is to let the user construct a dynamic model of a molecular (or other) network from biological knowledge, and allow a basic exploration of model dynamics and the effect of certain perturbations. In order to simplify model construction, a number of default settings are used that can subsequently be customized by the user, prioritizing simplicity and speed. The user first constructs a network in the form of a directed graph using a simple user interface, which indicates the causal dependencies of the network nodes. Our modeling framework of choice is that of dynamic models that are time-discrete, i.e., variables are updated in discrete time steps, and state-discrete, i.e., each variable can take on a finite number of possible states, currently up to five, and that number can vary across variables. The result is a finite (but conceivably large) space of possible system states (given in the form of a directed graph, with directed edges indicating state transitions). Each network node has attached to it a function that takes as input the states of all the nodes from which there is an incoming arrow, and provides as output the “next” state of the node. As a special case, each node could take on exactly two values, resulting in a Boolean network. Such a function can be specified in a number of ways, for instance through a Boolean function in the case of binary inputs. We have chosen the most simple and intuitive way, through the specification of a transition table that specifies the output for each possible input vector of states. Such a table is automatically generated as default, integrating the different inputs in an additive
fashion. The table can subsequently be customized, for instance, to use synergistic action instead of additive. Each row of such a table can be interpreted as a biological statement, e.g., “If A is high, B is low, and C is high at time $t$, then C will become low at the next time step,” representing the row [1 0 0 | 0] in the Boolean case. Thus, there is no need to learn any mathematical formalism to specify functions. The user is able to carry out basic computational “experiments,” namely to knock out network nodes and the arrows/interactions connected to those nodes. Finally, the user can analyze the model by computing all attractors, typically corresponding to different cellular phenotypes, and attractor basins, corresponding to the relative likelihood of that phenotype.

User Interface
The graphical user interface (GUI) guides the user interactively and dynamically through the modeling and analysis steps. Four arrow-shaped tabs are displayed along the top of the canvas rectangle in the natural order of constructing a model, setting up and carrying out computational experiments and analyzing the results. The transitions between the different modes can be done automatically according to the functional process of modeling activities or manually by clicking any arrow tab to open a functionality. The currently open mode is always indicated by a green color highlight on its tab, so the user knows in which mode is currently selected. We now describe these four modes in more detail. We will use the following small generic model as a running example, which can also be found on the PlantSimLab website:

$$A \rightarrow C \leftarrow B$$

where $A$ has two states (0, 1), $B$ has three states (0, 1, 2), and $C$ has four states (0, 1, 2, 3). Here, $A$ has an activating influence on $C$, and $B$ has an inhibiting influence.

Model Editor
This tab provides a canvas drawing area and a suite of graphical model editing tools for the user to draw a network model that is a graph-theoretic representation of the molecular network to be modeled. The user can create a node by clicking on the node icon in the toolbar and then clicking the location to place the node on the canvas. Then the user chooses the number of states for the node with state labels chosen from a predetermined menu or customized. Internally, the states get converted into numerical values, beginning with “0” for the first state in the table, up to “$n-1$,” where $n$ is the number of states (currently limited to five).

To create an edge, the user can click on an edge icon (with choices including “activate,” “inhibit” or “unspecified”) in the toolbar and then click the input and the target node successively. For an “activate” edge, a state transition table is created that captures the effect of states of the input node on the states of the output node. This table can then be modified by the user, similar to the table for an edge whose nature is unspecified. If a node has several input edges, their effects are combined into a comprehensive transition table called “Big State Transition Table” (BSTT), where the different inputs are integrated using an “additive” rule by default. That is, if a node receives two or more inputs, then the input values are added (e.g., if the edges are activating), respectively subtracted (e.g., if one or more nodes are inhibiting), depending on the edge tables at
each time step. We use the arithmetic convention that the resulting number cannot be lower than 0 or larger than \( n-1 \). Again, the user can customize this default choice. Several editing features simplify the management and editing of large tables.

**Calculation of state transitions.**

To show how PlantSimLab calculates the predetermined state transitions for a node, we created a very simple network model having only three nodes named “A” (2 states node), “B” (2 states node), and “C” (3 states node), where node A activates node C and node B inhibits node C (Figure 1.a)). The table in Figure 1.b is the big state transition table (BSTT) for node C. It displays all the possible combinations of the input node states in the current time cycle and their corresponding target node states in the next time cycle. Based on additive rules, PlantSimLab calculates the state of the target node \( C_t \) in the next time cycle, represented as \( C_{t+dt} \), for any possible combinations of the input nodes A, B, and C at time \( t \) as follows. The table entries in Figure 1.b for which B is equal to 0 give the effect of A alone on C. Likewise, the entries for which A is equal to 0 give the effect of B alone. The cumulative effect is assumed to be additive, in the sense that the right-hand column of the table is obtained as follows:

\[
C_{t+1} = A_t - B_t + C_t,
\]

subject to the constraint that the value is equal to 0, resp. 2, if \( A_t - B_t + C_t < 0 \) (integer arithmetic), resp. if \( A_t - B_t + C_t > 2 \).

Currently, PlantSimLab allows up to 5 states for a node, rendered as 0, 1, ..., 4. The exhaustive enumeration of all state transitions calculated using the same additive rule with adjustment for the lowest and the highest possible state values in this way. The BSTT opens automatically when the node is double-clicked (Figure 1.c). The user can use the predetermined choice, or change it by selecting a desired state from the dropdown box in the appropriate row of the BSTT (Figure 1.d).

A PlantSimLab network model can be thought of as a wiring diagram, with metadata attached, and the content of the canvas can be saved as an image file on the user’s local machine for inclusion in publications or presentations. When a network model is created in **Model Editor**, the user can explore the network dynamics of the unchanged model by running Dynamical Network Analysis algorithm with a click on short-cut button “go” provided in the **Model Editor** toolbox.
**Figure 1.** The wiring diagram of a simple network model containing three nodes (a). A table to show the calculation of all possible state transitions for a target node C in the network model (b). Double clicking on a node on the Model Editor canvas highlights the node in the back and opens up the Big State Transition Table (BSTT) for the node (c). The user can use the predetermined choice, or change it by selecting a desired state from the dropdown box in the row (d).

**Experimental Setup**

This tab provides a canvas drawing space and a menu of experiments, which currently consists of the possibility to simulate the knock-out of one or more nodes.

The default setting for a knocked-out node is constant equal to the node state corresponding to “0” for all simulation time steps. This can be done for several nodes simultaneously. The user can customize this feature and set the state of a “perturbed” node constant equal to any chosen state. In this way, one can also simulate knock-down and overexpression of one node or a combination of these for more nodes simultaneously. Knocking out a node or reversing a knockout can be done on a single node using a context menu popped up at the right-click on a node to knock out or undo (Figure 2.a), or on multiple nodes at once using the Experimental Setup Table (Figure 2.b), which provides a knockout button for each node so the user can select the multiple nodes to knock out or undo the knockout all at once.
For an experiment, the user can perturb a model by knocking out nodes and the associated edges. The user can do this on a single node using a context menu popping up after right-clicking on a node to knock out or undo (a), or using the Experimental Setup Table. The initial state of the node can be set to the desired state in this tab using Experiment Setup Table (b). A knocked-out node has a X mark in red through it (c).

**Model Analysis**

For either the “wild-type” or experimentally perturbed model, an analysis consists of the computation of the collection of attractors of the model, both steady state and periodic, as well as the size of the individual attractor basins, reported as a percentage of the overall state space. This is done using exhaustive simulation of all state transitions.

**Results Viewer**

This tab is designed to display the network analysis results using various HTML forms, styles, and devices to deliver the output information in an intuitive and easy to understand format. The summary table (ST) shows all attractors of the dynamic network and their basins of attraction. The magnitudes of the node states in the attractor are presented by their numerical values as well as in a heat-map-style color scheme to enable a quick grasp of the information about the attractor. The relative contributions of each attractor basin to the entire state space are given as a percentage. A pie-chart combining all components into a compact view also enables a quick visual gauge of the relative contributions of each attractor. Clicking on a particular row in the ST or a pie segment from the pie-chart opens up a separate window and displays the detailed information about the selected subset of the state space, including the state space graph of the subset. To create a more intuitive and easy to understand space state graph, we used HTML style bar size and color coding for presenting the state of a node (a cell for a low state node in the graph is filled with a small bar in yellow, for a high state node with a full size bar in purple, and for a medium state node with a half size bar in gray (Figure 3.d). For better visibility, the user can control the direction of the state space graph display horizontally or vertically using toggle buttons. The user will notice that the state transition of any knock-out node remains constant equal to the state corresponding to 0. The ST can be conveniently saved into an Excel file on the user's local machine.
Figure 3. The summary table (ST) displays all attractors and the attractor basin in the system for the running example model with three nodes introduced above. The HTML table uses heat-map style color scheme to display the steady states of nodes. The ST in the left side is generated for an unchanged model (a) and the ST in the right side is for a perturbed experimental model (b). For all perturbed models, PlantSimLab displays a CAUTION message to inform of the potential existence of other steady states or limit cycles than the displayed (b) (see below for an explanation). The entire state space graph is drawn using different colors and box sizes to make the state space graph more visually intuitive and informative (c). A subset of the state space (d) can be also drawn for further study of the simulation of a network component selected with a click on a row corresponding to a particular component row in the ST, on a component piece in the pie-chart (e), or on a component entry from the Component Summary drop-down box provided in the Results Viewer toolbox.

Known issues/limitations: Due to space and computation time limitations, PlantSimLab does not draw the state space graph if the size of the state space exceeds 2600 nodes. Furthermore, the current algorithm used for Dynamical Network Analysis was not built to handle models with knocked-out nodes; rather, it automatically enumerates all possible state transitions of all nodes by default. This limits the functionality of the network perturbation feature. To partially resolve the issue, PlantSimLab currently excludes the output display of those steady states or limit cycles where the knocked-out nodes are not in their lowest state. As a result, for a perturbed
network only some of the steady states and none of the limit cycles are generated. The user is warned about this through a warning message whenever a perturbed network analysis is performed. The message advises the user to instead simulate a node knockout by setting the node state to low/off in the BSTT.

**YouTube tutorials:** We created three YouTube tutorial videos to provide instructions on how to create, perturb, and analyze a network model and view the network analysis results in PlantSimLab for hypothesis generation. Tutorial #1 shows how to create nodes and edges and configure them interactively using special tables for building a graphical network model in *Model Editor tab*. Tutorial #2 shows how to perturb a model with knocking out nodes in *Experimental Setup tab*. Tutorial #3 explains how to visually inspect the dynamical network analysis results in many different forms in the *Results Viewer tab*. All instructions in the tutorials are easy to follow and it takes only 5 to 11 minutes for each video. The three PlantSimLab YouTube videos are available through the following links:

- PlantSimLab Tutorial #1 - Model Building ([https://www.youtube.com/watch?v=97qa7rWYeIU](https://www.youtube.com/watch?v=97qa7rWYeIU))
- PlantSimLab Tutorial #2 – Experimental Setup ([https://www.youtube.com/watch?v=3sFwsxmJ-Qg](https://www.youtube.com/watch?v=3sFwsxmJ-Qg))
- PlantSimLab Tutorial #3 – Network Analysis Results Viewer ([https://www.youtube.com/watch?v=aI96BQF1eC8](https://www.youtube.com/watch?v=aI96BQF1eC8))

**DEVELOPMENT**

**Software Components**

We developed PlantSimLab as a client-server-based web application running on Apache2 on a dedicated server and supporting most modern web browsers on any platform. The user interface on the client was developed using HTML with significant components of JavaScript and AJAX to enhance the user experience. On the server side we used PHP scripting and an open source MySQL database for model repository management. The use of Google sign-in authentication for the user login promotes user convenience and reduces the burden of user profile management for the software.
Figure 4. The software components (a) and the software workflow (b) of PlantSimLab, highlighting the steps for modeling, analysis, and use. PlantSimLab communicates with the model database repository to load and save the user models. To perform network analysis, PlantSimLab runs Dynamical Network Analysis algorithm locally installed or that in the Computational Network Analysis workflow via the back-end pipeline (b).

To create an intuitive, well-designed, and frustration-free user interface, we applied design principles similar to Shneiderman’s ‘eight golden rules of interface design’ [11].
1. **Consistency** is woven into the design of all tabs, toolbars, tables and dialogues.
2. **Shortcut** buttons are provided in the toolbars to allow quick and easy execution of network analysis.
3. **Visual feedback** can be seen when a node or an edge becomes “highlighted” as the user clicks on them. The wiring diagram can be seen enlarged, shrunk, or moved physically on the screen when the user resizes or moves the diagram using the mouse.
4. **Informative screen** shows what step the user is currently in.
5. **Error or exception handling** provides “gentle” messages if an error or exception case occurs.
6. **Easy reversal of actions** is allowed when the user wants to undo a knock-out action.
7. **Support internal locus of control** rule is applied so navigation and task activation should always be clear and well-marked. A user should never be wondering “How did I get to this screen?” or “What do I need to press to do any task?”
8. **Reduce short-term memory load.** The limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be consolidated, window-motion frequency be reduced [11]. The four main tabs of PlantSimLab were designed with consideration of efficient memory load and functional process.

A Use Case.
To demonstrate the use of PlantSimLab we implement and analyze the model from Espinosa-Soto et al. [14]. This paper follows several other investigations into the gene networks driving cell fate determination in the model organism *Arabidopsis thaliana*. The authors focus on the question of robustness of morphological pattern development, in particular floral organ cell fate determination. Experimental studies led to the development of the ABC combinatorial model of gene expression states that predicts the identity of floral organ primordia, which has guided many experimental studies. The model presented in [14] is based on a more complete understanding of the genetic components and interactions involved, resulting in model steady states that are coherent with experimental data.

The model consists of 15 nodes, eight of which can assume two states, e.g. ON/OFF
(FT, EMF1, SEP, AP2, WUS, UFO, CLF, and Lug) and seven can assume three states, e.g. LOW/MEDIUM/HIGH (LFY, AP1, FUL, TFL1, AG, AP3, and PI). The wiring diagram of the model generated by PlantSimLab is presented in Figure 5, and is identical to Figure 4 in [14]. In [14], the logical rules for each node are provided in table form for each node, very similar to PlantSimLab’s transition tables. We implemented in PlantSimLab the proposed wild-type network and the loss-of-function ap2 mutant. The simulation of the wild-type network generated a total of 40 steady states and no other cycles (Figure 6). Among them were the 10 steady states recovered in [14], based on the 139,968 initial conditions considered there (the total number of possible initial conditions is 559,872). The simulation of the ap2 mutant network returned 28 steady states and no other cycles (Figure 7), with the seven steady states simulated in [14] present among them. We note that both simulations considered all possible initial conditions and, as a result, were able to generate all network steady states, with some of the additional ones having relatively large basins of attraction (up to 22% of all states). It took PlantSimLab only seconds to complete the simulations, and the additional steady states that were discovered might provide valuable additional information other possible phenotypes of the cells considered.

**Figure 5:** Wiring diagram of the network, identical to Figure 4 in [14].
### Component List

#### Component Distributions

| Component | Type | Size |
|-----------|------|------|
| Component 1 | EMF(2) | 1332 (0.23%) |
| Component 2 | LFY(3) | 936 (0.16%) |
| Component 3 | AP1(3) | 1332 (0.23%) |
| Component 4 | AT2(2) | 3044 (0.54%) |
| Component 5 | SEP(2) | 3960 (0.70%) |
| Component 6 | LFY(3) | 1332 (0.23%) |
| Component 7 | AP1(3) | 1332 (0.23%) |
| Component 8 | AT2(2) | 3044 (0.54%) |
| Component 9 | SEP(2) | 3960 (0.70%) |

### Figure 6

The list of steady states and component sizes from the wild-type network simulation.
**Figure 7**: The list of steady states and component sizes from the *ap2* knock-out network simulation.

**Conclusions**

In this work, we present the web-application tool PlantSimLab, a simple, intuitive software tool for creating dynamic network models, doing experimental simulations, and visualizing network analysis results using a variety of formats and dynamic layouts. It provides a computational laboratory for biological scientists to generate novel experimental hypotheses. It is designed to be usable after completing a brief online interactive tutorial that explains the basic input, output, and functionalities of the software. It was our goal to make all features of PlantSimLab’s interface as intuitive for life science researchers and self-explanatory as possible. Future development of the software tool includes a simple natural language parser that allows users to provide English language sentences with prescribed syntax, such as “A [interacts with] B,” which are then automatically translated into network components. The YouTube tutorials will be extended to cover more advanced technical topics.

**Availability and requirements**

Project Name: PlantSimLab
Project home page: [www.plantsimlab.org](http://www.plantsimlab.org)
Operating System(s): Linux, Windows, MacOS
Programming language: HTML5, JavaScript, PHP
Any restriction to use by non-academics: None

ACKNOWLEDGEMENTS
We thank J. Brunson, S. Lee, and B. Klahn for their valuable user input and other feedback.
Funding: National Science Foundation Award DBI-1146819 is gratefully acknowledged.
Conflict of Interest: none declared.

REFERENCES
1. Elton, C., *Eppur si mouve*. J. Animal Ecology, 1935. 4(1): p. 1480150.
2. Lucas, M., L. Laplaze, and M.J. Bennett, *Plant systems biology: network matters*. Plant Cell Environ, 2011. 34(4): p. 535-53.
3. Dinh, J.L., E. Farcot, and C. Hodgman, *The logic of the floral transition: Reverse-engineering the switch controlling the identity of lateral organs*. PLoS Comput Biol, 2017. 13(9): p. e1005744.
4. Garcia-Gomez, M.L., E. Azpeitia, and E.R. Alvarez-Buylla, *A dynamic genetic-hormonal regulatory network model explains multiple cellular behaviors of the root apical meristem of Arabidopsis thaliana*. PLoS Comput Biol, 2017. 13(4): p. e1005488.
5. Oles, V., A. Panchenko, and A. Smertenko, *Modeling hormonal control of cambium proliferation*. PLoS One, 2017. 12(2): p. e0171927.
6. Velderrain, J.D., J.C. Martinez-Garcia, and E.R. Alvarez-Buylla, *Boolean Dynamic Modeling Approaches to Study Plant Gene Regulatory Networks: Integration, Validation, and Prediction*. Methods Mol Biol, 2017. 1629: p. 297-315.
7. Helikar, T., et al., *The Cell Collective: toward an open and collaborative approach to systems biology*. BMC Syst Biol, 2012. 6: p. 96.
8. Chaouiya, C., A. Naldi, and D. Thieffry, *Logical modelling of gene regulatory networks with GINsim*. Methods Mol Biol, 2012. 804: p. 463-79.
9. Mussel, C., M. Hopfensitz, and H.A. Kestler, *BoolNet--an R package for generation, reconstruction and analysis of Boolean networks*. Bioinformatics, 2010. 26(10): p. 1378-80.
10. Dussaut, J.S., et al., *GeRNNet: a gene regulatory network tool*. Biosystems, 2017. 162: p. 1-11.
11. Forbes, A.G., et al., *Dynamic Influence Networks for Rule-Based Models*. IEEE Trans Vis Comput Graph, 2018. 24(1): p. 184-194.
12. Livigni, A., et al., *A graphical and computational modeling platform for biological pathways*. Nat Protoc, 2018. 13(4): p. 705-722.
13. Wu, H., et al., *MUFINS: multi-formalism interaction network simulator*. NPJ Syst Biol Appl, 2016. 2: p. 16032.
14. Espinosa-Soto, C., P. Padilla-Longaria, and E.R. Alvarez-Buylla, *A gene regulatory network model for cell-fate determination during Arabidopsis thaliana flower development that is robust and recovers experimental gene expression profiles*. The Plant Cell, 2004. 16: p. 2923-2939.