Published in final edited form as:

Cell Syst. 2018 August 22; 7(2): 180–184.e4. doi:10.1016/j.cels.2018.06.004.

The Cell Cycle Browser: an interactive tool for visualizing, simulating, and perturbing cell cycle progression

David Borland[#1], **Hong Yi**[#1], **Gavin D. Grant**[2,3], **Katarzyna M. Kedziora**[4], **Hui Xiao Chao**[4,5], **Rachel A. Haggerty**[4,5], **Jayashree Kumar**[5], **Samuel C. Wolff**[4], **Jeanette G. Cook**[2,3], and **Jeremy E. Purvis**[3,4,5,†]

[1]Renaissance Computing Institute University of North Carolina, Chapel Hill, 120 Mason Farm Road, Chapel Hill, NC 27599-7264

[2]Department of Biochemistry and Biophysics University of North Carolina, Chapel Hill, 120 Mason Farm Road, Chapel Hill, NC 27599-7264

[3]Lineberger Comprehensive Cancer Center University of North Carolina, Chapel Hill, 120 Mason Farm Road, Chapel Hill, NC 27599-7264

[4]Department of Genetics University of North Carolina, Chapel Hill, 120 Mason Farm Road, Chapel Hill, NC 27599-7264

[5]Curriculum for Bioinformatics and Computational Biology University of North Carolina, Chapel Hill, 120 Mason Farm Road, Chapel Hill, NC 27599-7264

[#] These authors contributed equally to this work.

Abstract

The cell cycle is driven by precise temporal coordination among many molecular activities. To understand and explore this process, we developed the Cell Cycle Browser (CCB), an interactive web interface based on real-time reporter data collected in proliferating human cells. This tool facilitates visualizing, organizing, simulating, and predicting the outcomes of perturbing cell cycle parameters. Time-series traces from individual cells can be combined to build a multi-layered timeline of molecular activities. Users can simulate the cell cycle using computational models that

† Lead Contact: Jeremy Purvis, Genetic Medicine Building 5061, CB#7264, 120 Mason Farm Road, Chapel Hill, NC 27599-7264, jeremy_purvis@med.unc.edu.

AUTHOR CONTRIBUTIONS

D.B. and H.Y. developed the user interface and server-side software, contributed to the conceptual design of the CCB, and wrote portions of the manuscript. G.G. conceptualized the virtual experiments and contributed live-cell data. K.K. and H.X.C contributed live-cell data, performed image analysis, and standardized data formatting. R.A.H. developed the logo and contributed to visualization and image analysis. J.K. developed SBML models to simulate cell cycle progression. S.W. performed time-lapse imaging and contributed live-cell data. J.G.C. proposed virtual experiments and improvements to the user interface and contributed live-cell data. J.E.P. conceived of the CCB and wrote portions of the manuscript.

DECLARATION OF INTERESTS

There are no competing interests.

CONTACT FOR DATA SHARING

Please contact Jeremy Purvis at jeremy_purvis@med.unc.edu with requests and inquiries.

capture the dynamics of molecular activities and phase transitions. By adjusting individual expression levels and strengths of molecular relationships, users can predict effects on the cell cycle. Virtual assays, such as growth curves and flow cytometry, provide familiar outputs to compare cell cycle behaviors for data and simulations. The CCB serves to unify our understanding of cell cycle dynamics and provides a platform for generating hypotheses through virtual experiments.

Summary

Borland, Yi, and colleagues develop an interactive website offering visualization, simulation, and virtual experiments to study cell cycle progression at the molecular level.

Keywords

cell cycle; live-cell imaging; single-cell dynamics; computational modeling; scientific visualization

The cell-division cycle—the process by which cells replicate their DNA to form two daughter cells—is central to nearly every facet of the life sciences including disease research and industrial biotechnology. A long-term challenge for the field is to determine precisely how the cell cycle is coordinated in time and how that coordination changes under different environmental conditions. A growing number of recent studies have monitored various molecular entities in single cells over the course of the cell cycle including the activities of cyclin-dependent kinases (Spencer et al., 2013), stress response factors (Loewer et al., 2010), DNA damage repair foci (Arora et al., 2017), and DNA replication (Barr et al., 2017, Chao et al., 2017). A high-resolution timeline that accounts for the interrelationships among cell cycle entities would be a transformative tool for many basic and applied fields of research. Such a map would predict how future changes—such as a newly discovered mutation, environmental exposure, or drug—are likely to affect cell proliferation and genome stability. Here, we describe a web-based tool, the Cell Cycle Browser (CCB), for visualizing, simulating, and analyzing the timeline of molecular events during cell cycle progression (https://cellcycle.renci.org). This article provides a high-level overview of the CCB. A more thorough tutorial of the tool is available at https://github.com/RENCI/CellCycleBrowser/wiki/tutorial.

CCB Workspace.

The information displayed by the CCB is organized into *workspaces*. A workspace consists of a collection of *datasets*—experimentally measured, single-cell traces of molecular activities—and cell cycle *models*—files describing the kinetics relationships between molecular activities and the rates of cell cycle progression. Cell cycle models can be simulated and compared to experimental data. In the data selection section of the CCB graphical user interface (GUI) (Figure 1–1), the user can select from a list of available workspaces. After selecting a workspace, the datasets and first model included in that workspace are automatically loaded. The user can add additional datasets, remove any currently loaded datasets, or select a different model. Users can upload their own data into a "playground" workspace and create their own SBML model. Users can also download all CCB data and models in a single zip file.

Each dataset contains a time-series of measurements for any number of molecular species captured across multiple individual cells. Each single-cell time-series, or trace, is derived from the analysis of segmented images of proliferating human cells expressing a fluorescent reporter. Therefore, each species may contain multiple features such as reporter intensity, number of foci, nuclear area, or the spatial position of the cell. The available features for each dataset can be selected from the dropdown menu, and are displayed in the Data Browser. The user can select a single model from the dropdown model list, or choose "None" if interested in viewing only the experimental datasets. Finally, the user can select which analysis plots to display including a time-series plot of experimental and simulated data; a growth curve plot for each dataset/simulation; and a simulated analytical flow cytometry plot of DNA synthesis vs. DNA content (FACS) for the current simulation, or any datasets that contain phase information.

Data Browser.

One of the primary uses of the CCB is that it provides a common time axis to visualize multiple time-series traces from individual cells representing diverse molecular activities. The center section of the CCB provides a heat-map representation of single-cell traces from the current experimental data and/or simulated output (Figure 1–2). To capture a single cell cycle, each cell's trace is excised from the telophase event that generated that cell until its own telophase (Davis and Purvis, 2015). A separate panel, or *track*, is drawn for each selected image feature across all molecular species in a particular data *source*. A data source refers to an experimental dataset, which may contain multiple reporters and image features, or the current model simulation. Each track consists of a collapsible area showing each individual cell trace with value mapped to color (white: low, black: high). Color scaling, which performs unit-normalization on the time-series values, can be performed per track, or per trace. An average trace for each track is displayed above the single-cell traces. Average traces are generated by scaling the length of each individual trace to match the average length of all traces per track and then performing nearest-neighbor sampling for each sample point in the average. Traces can be aligned to the beginning (first telophase), the end (second telophase), or individually stretched to align each trace to the beginning and end of the cell cycle.

The heat-map visualization facilitates differentiation of individual traces while still supporting analysis of different temporal patterns (Correll et al., 2012, Albers et al., 2014). Users can perform hierarchical clustering to automatically classify cells into different groups according to their single-cell dynamics. In addition to molecular activities, the CCB draws tracks showing the locations of individual cell cycle phases (G1, S, G2-M) both for the simulated data as well as for any datasets that contain phase information. Tracks can be sorted by type (i.e., data or phase), species, image feature, or data source. The user can also arrange tracks by dragging and dropping them individually. Buttons to the left of each trace control whether or not to show that trace in the time-series analysis plot, and colored icons in the track header indicate the color for that data source/track in any displayed *Analysis Plots* (described below).

Analysis Plots.

Three analysis plots are available to provide additional information about the experimental and simulated data (Figure 1–3). A time series plot shows each individual or average cell trace—selected using buttons in the data browser tracks—as a line plot. Two or more time series plots from the same source have the same base color, with different shades of that color for each track from the data source. These colors are also indicated by the color icons in each track; the data source color is used for the other analysis plots. Linked highlighting between traces in the time series plot and the data browser is provided. A growth curve plot shows fold change growth per day for each data source, based on the average length of the traces for that track. Flow cytometry plots (DNA synthesis on the y axis and DNA content on the x-axis) are simulated for each data source containing phase information. Each data point represents a simulated cell, with phase generated probabilistically based on the average phase lengths for that data source (Toettcher et al., 2009). Each cell is colored based on the local density of neighboring cells. Users can download analysis plots and the model visualization described in the next section in Scalable Vector Graphics (SVG) format.

Model Visualization.

The left section of the CCB contains a network visualization of the cell cycle model (Figure 1–4) as well as controls for adjusting the model and simulation parameters (Figure 1–5). In a model, each molecular activity is referred to as a *species*—a pool of entities that are assumed to be indistinguishable, located in the same cellular compartment, and involved in the same molecular reactions (Hucka et al., 2003). Each cell cycle model specifies: (1) the abundance of each species (e.g., activity of Cdk2; number of 53BP1 foci); (2) the influence of each species on each phase transition (e.g., influence of Cdk2 activity on the G1/S transition; inhibitory effect of 53BP1 damage foci on G2/M transition); and (3) the influence of each species on every other species for each phase (e.g., an increase in 53BP1 foci during G2 induces expression of p21).

In the network diagram, each species is represented as a circular node whose size is proportional to the species' abundance (specifically, the fold change compared to the initial value specified in the model). A separate species diagram is drawn for each phase since interactions may vary throughout the cell cycle. Links between two species or between a species and a particular phase transition indicate the interaction between those entities, color-mapped from blue (inhibiting) through grey (no influence) to red (activating), with magnitude mapped to width. Selecting a particular phase in the visualization activates the control tabs for that phase in the interaction panels, described below.

Model Simulation and Perturbation.

A set of computational models describing cell cycle progression have been developed for different cell types and molecular activities. These models can be simulated under the default conditions or after adjusting model parameters. Simulation parameters can be adjusted using sliders that control: a) number of cells to simulate, b) species expression levels, c) species to phase interaction parameters, and d) species-to-species interaction parameters per phase. Interaction slider handles are colored to match the links in the network diagram visualization. Users can also upload or create their own SBML models.

When the simulation button is pressed, the StochPy library (Maarleveld et al., 2013) is used to perform cell cycle simulations on the server via the stochastic simulation algorithm (SSA) (Gillespie, 1977, Gillespie, 2007). A customized StochPy simulation engine terminates the simulation when the last phase of the cell cycle (e.g., M-phase) is completed in order to simulate exactly one complete cell cycle per simulation. Because some simulation parameters can cause extremely slow cell cycle simulation (e.g., high expression of p21), a red cancel button is available so users can cancel the run if the simulation is taking an excessively long time. In addition, the current cell and phase of simulation is displayed in a progress bar. After simulation, users can calculate the fit of simulation results to experimental data in a panel located below the simulation buttons (Figure 1–6).

In summary, the Cell Cycle Browser (CCB) (https://cellcycle.renci.org) is a web-based tool for exploring, simulating, and perturbing the human cell cycle via an interactive graphical user interface. It was developed in the spirit of the UCSC Genome Browser (https://genome.ucsc.edu), which provides a graphical view of genome sequence data along the physical coordinates of the chromosomes. Unlike the Genome Browser, however, the primary axis of the CCB is time. In its most basic functionality, the CCB displays time series data representing various molecular activities over the course of a single cell cycle. Single-cell traces from time-lapse fluorescent microscopy experiments - representing different biological activities from multiple cell lines - may be aligned in time to examine the temporal sequence of cell cycle events and explore new relationships among cell cycle events. In addition, the CCB enables a user to simulate a mechanistic model of cell cycle progression and align the model output with experimental data in time to better understand and predict cell cycle behaviors. Future improvements to the CCB will include: (1) the ability to create customized, password-protected workspaces for individual users that allows them to create private workspaces that they can choose to make public later; (2) data from additional organisms; (3) a portal for submitting raw data for review to be available for visualization as tracks; (4) automated imputation of phase transition locations based on single-cell traces; and (5) the ability to load, simulate, and compare multiple computational models.

STAR*METHODS

KEY RESOURCES TABLE

RESOURCE	SOURCE	IDENTIFIER
Sample Data		
All live-imaging data sets and SBML models	(multiple)	https://cellcycle.renci.org/download_all_data/

RESOURCE	SOURCE	IDENTIFIER
Software and Algorithms		
CCB tool software source code repo	This paper	https://github.com/RENCI/CellCycleBrowser/
Docker	https://www.docker.com/	v1.7.1
Django Python web framework	https://www.djangoproject.com/	v1.10
Python	https://www.python.org/	v2.7.11
PostgreSQL database	https://www.postg resql.org/	v2.6.1
RabbitMQ message broker	https://www.rabbitmq.com/	v1.4.9
Celery distributed task queue	http://www.celeryproject.org/	v3.1.23
StochPy stochastic modeling in Python	http://stochpy.sourceforge.net/	v2.3
libSBML Python library	http://sbml.org/Software/libSBML/docs/python-api/	v5.15.0
Bootstrap	http://getbootstrap.com/	v3.3.6
jQuery	https://jquery.com/	v1.12.4
d3	https://d3js.org/	v4.10.0
d3-contour	https://github.com/d3/d3-contour	v1.2.0
d3-scale-chromatic	https://github.com/d3/d3-scale-chromatic	v1.2.0
flux	https://github.com/facebook/flux	v3.1.3
hclusterjs	https://github.com/cmpolis/hcluster.js	v1.3.1
keymirror	https://github.com/STRML/keyMirror	v0.1.1
object-assign	https://github.com/sindresorhus/object-assign	v4.1.1
react react-dom	https://facebook.github.io/react/	v 16.2.0
prop-types	https://github.com/facebook/prop-types	v15.6.1
science	https://github.com/jasondavies/science.js/	v1.9.3
seedrandom	https://github.com/davidbau/seedrandom	v2.4.3
simple-statistics	https://simplestatistics.org/	v5.2.1
babel-core babel-cli	https://github.com/babel/babel	v6.26.0
babel-loader	https://github.com/babel/babel	v.7.1.4
babel-preset-env	https://babeljs.io/docs/plugins/preset-env/	v1.6.1
babel-preset-react	https://babeljs.io/docs/plugins/preset-react/	v6.24.1
html-webpack-plugin	https://webpack.js.org/plugins/html-webpack-plugin/	v3.1.0
webpack	https://webpack.js.org/	v4.2.0
webpack-cli	https://webpack.js.org/api/cli/	v2.0.13
webpack-dev-server	https://webpack.js.org/configuration/dev-server/	v3.1.1
Other		
Website for CCB portal	This paper	https://cellcycle.renci.org
Tutorial for CCB tool	This paper	https://github.com/RENCI/CellCycleBrowser/wiki/Tutoria
CCB installation guide for developers	This paper	https://github.com/RENCI/CellCycleBrowser/blob/master/READ

METHOD DETAILS

Server handling of data and model simulation

JSON was used as the data interchange format to pass AJAX updates between the client and the server. Figure S1 shows a pipeline of detailed data flow to illustrate how data flows between the client and the server via AJAX when the page initially loads and at different requests from user interactions with the browser.

When the main page initially loads, an AJAX request is sent from the client and the server responds with an array of key-value pairs for all workspaces (collections of datasets and models) in JSON format. Key-value pairs for each workspace include name and description of the workspace along with an array of key-value pairs for all datasets and models included in the workspace. The key-value pairs for each dataset or model include name, description, and the name of the file stored on the server. The client also requests from the server via AJAX a list of all datasets and a list of all models as needed to populate the user interface.

Each experimental dataset is stored on the server as a series of time-stamped values of features for one or more species in CSV format, with metadata included in a header section. Each model input file is stored on the server in SBML (Systems Biology Markup Language) format. When a dataset is selected, the client sends AJAX requests to the server to retrieve time-stamped data for the selected dataset for visualization. When a model is selected, a corresponding AJAX request is sent and the server extracts initial parameter data from the model and sends the data in JSON format to the client for display in the model control panel and to construct and display the map visualization. These model initial parameter data include species expression levels, species-phase interactions, and species-species interactions per phase; all can be adjusted from the user interface before triggering a model simulation run.

When a model simulation run is triggered from the user interface, the client sends the simulation request to the server via AJAX, including all input parameters and the number of cells/trajectories to simulate. The server performs the stochastic simulation algorithm (SSA) as an asynchronous task in a separate worker thread on the web server without blocking the main web thread. Utilizing multithreading and asynchronous model simulation runs makes the CCB web tool responsive, scalable, and performant even with large numbers of simultaneous web requests for model simulation runs. When the simulation is complete, the server sends the simulation result to the client as an AJAX response so that the simulation result can be visualized and analyzed to compare with experimental datasets.

Model Specification

Models were specified using the Systems Biology Markup Language (SBML) (Hucka et al., 2003). We used LibSBML (Bornstein et al., 2008) to write SBML models using the following user-specified parameters:

1. The name of the output file (e.g., “model.xml”).
2. k and λ values for each cell cycle phase using the Erlang model in Chao et al. (Chao et al., 2017).

3. A list of *n* species names, S.
4. A list of *n* initial conditions, X0. Values must be >= 0.
5. An *n*-by-*m* array of species production rates in each phase, B, where *m* is the number of phases. Values must be >= 0.
6. An *n*-by-*m* array of species degradation rates in each phase, A. Values must be >= 0.
7. An *n*-by-*m* array of species-phase interaction strengths, P. These values indicate how each species increases or decreases the rate of progression through a particular phase. Values can be positive, negative, or zero. A value of 0 does not change the rate of progress. Values less than 0 slow progress and values greater than 0 increase the rate of progress.
8. An *n*-by-*n*-by-*m* array of species-species interactions for each phase, Z. Each value Z_{ijk} represents the strength of the influence of species *i* on species *j* during phase *k*. Values can be positive, negative, or zero. Positive values indicate promoting interactions; negative values indicate negative interactions. A value of 0 indicates no interaction.

Using these values, LibSBML generates an XML file using the following steps:

1. Generate an “invisible” species for each phase (e.g., “G1”) and subphase (e.g., “G1_01”, “G1_02”,…). Phase and subphase values can be either zero or one. Only one subphase can be one; all others must be zero.
2. Generate an assignment rule so that each phase is the sum of all of its subphases. In this way, each phase/suphase variable is an indicator of the current phase/subphase.
3. Generate reactions to transition between each phase (e.g., consuming G1_01 and producing G2_02 at rate λ_{G1})
4. Generate each species in list S and set its initial condition to values in X0.
5. For each species, S_i, generate a production reaction: null -> S_i. Set the rate law to: Bi1*(G1) + Bi2*(S) + Bi3*(G2), where Bi1 refers to the production term for the ith species for the G1 phase (the first column in the array, B). Since G1, S, and G2 are indicator variables, this ensures that Si is produced at rate Bi1 only when the cell is in G1.
6. For each species, Si, generate a degradation reaction: Si -> null. Set the rate law to: Ai1*(G1)*Si + Ai2*(S)*Si + Ai3*(G2)*Si, where Ai1 refers to the degradation term for the ith species for the G1 phase (the first column in the array, A). Since G1, S, and G2 are indicator variables, this ensures that Si is degraded at rate Ai1 only when the cell is in G1.
7. For each element Pij in array P, add Pij to each of the reactions for phase transition j. For example, if Pij represents the effect of 53BP1 on S phase, then all of the S phase transition reactions (e.g., S1 -> S2) should have rate law: l_S *

power(1+S1, Pij). Here, l is the rate already specified in the first 6 arguments that define basal transition rates. Pij will speed up or slow down this rate.

8. For each non-zero element Zijk, generate a reaction. If Zijk > 0, generate a production reaction: null -> Sj with rate law: Zijk * Si. If Zijk < 0, generate a degradation reaction: Sj -> null with rate law: Zijk * Si * Sj.

Pre-defined model parameters for species production, degradation, and interactions were adjusted empirically to resemble average single-cell traces for each species.

Client/GUI architecture

The client was built with JavaScript and HTML using the React library for creating custom GUI components, Bootstrap for GUI styling, and D3 for creating the different SVG-based visualization elements. The Flux architecture is used to ensure consistent dataflow.

Model parameters

There are two types of model parameters that can be adjusted by the user via sliders: species expression levels and interactions between species and phases, and between species. The expression level for each species in the initial model is the basal level for that protein. As such, these may differ between species. The slider controls for the expression level impart a fold-change on this initial expression level, with each slider initially at a value of 1x. Sliding to the right imparts a fold increase, up to 32x, and to the left a fold decrease down to 1/32x of the initial value, which essentially removes the species via virtual “knock down”. The interaction sliders describe the exponent in the rate equations for each slider. These sliders range from −10 to 10, where more negative numbers apply an increasingly inhibiting interaction, and more positive numbers apply an increasingly promoting interaction. In the model, the species expression level is controlled by altering the date of degradation for that species across all phases.

Averaging traces

To calculate the average trace for each track, the average time steps across all traces in that track is calculated and used to generate sample points up to the average trace length. Each trace is then stretched temporally to match the average trace length, and nearest-neighbor interpolation is used to average across all traces for each sample point.

Hierarchical clustering

Hierarchical clustering of the traces in each track is performed by calculating the average difference per sample point average difference between traces at each sample point using the current rescale and alignment settings. The hcluster.js library is used, which employs agglomerative hierarchical clustering.

Growth curve calculation

Growth curves for each data source (experimental dataset or simulation) are shown in the Growth Curve Analysis Plot. The growth curves are calculated from the average trace length in that data source with the formula $y = 2^{t/l}$ where is the fold change in cell number, t is the independent variable of time on the x-axis of the growth curve, and *l* is the average duration of the cell cycle for a group of single-cell traces.

Cell cycle analysis plots

A virtual flow cytometry cell cycle analysis was performed for each data source containing phase information and displayed in the Cell Cycle Analysis Plot. Separate scatter plots of the steady-state of a large number of cells are generated by resampling a kernel-density estimate of flow cytometry from an experimental dataset, based on the phase lengths in the given data source. The phase probability distributions for each data source are calculated based on the appropriate formulas in (Toettcher et al., 2009). First, we calculate **τ**phase, the normalize length of each phase, by dividing the average length for that phase by the average cell cycle length for that data source. The proportion of cells, P_{phase}, in each phase are then:

$$P_{G1} = 2 * (1 - 2\hat{}(- \boldsymbol{\tau}_{G1})),$$

$$P_{G2} = 2\hat{}(- \boldsymbol{\tau}_{G2}) - 1,$$

$$P_S = 1 - (P_{G1} + P_{G2}).$$

These probabilities are used to generate the phase for each simulated cell, and then resampling of the kernel density estimate for that phase is used to generate an (x, y) point in the resulting scatterplot.

Model fit

The model fit between a user-selected simulation track and experimental data track is calculated as one minus the difference between the population-averaged, stretched, simulated traces and the population-averaged, stretched, experimental traces—averaged over individual time points. The exact meaning of the units of this value is difficult to interpret; however, it provides a consistent way to compare the goodness of fit across different simulation runs and data sets. Alternatively, users can choose to score their fit based on the correlation of population-averaged, stretched traces.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

ACKNOWLEDGEMENTS

We thank Sabrina Spencer, Mansi Arora, Sam Cooper, Alexis Barr, and Chris Bakal for contributing published live-cell data to the browser. This work was supported by a Medical Research Grant from the W.M. Keck Foundation (J.E.P. and J.G.C.) and NIH pre-doctoral fellowships F30-CA213876 (H.X.C.) and F31-HL134336 (R.A.H.)

REFERENCES

Albers D, Correll M and Gleicher M (2014). Task-driven evaluation of aggregation in time series visualization. Proc SIGCHI Conf Hum Factor Comput Syst, 551–560. [PubMed: 25343147]

Arora M, Moser J, Phadke H, Basha AA, and Spencer SL (2017). Endogenous replication stress in mother cells leads to quiescence of daughter cells. Cell Rep 19, 1351–1364. [PubMed: 28514656]

Barr AR, Cooper S, Heldt FS, Butera F, Stoy H, Mansfeld J, Novak B, and Bakal C (2017). DNA damage during S-phase mediates the proliferation-quiescence decision in the subsequent G1 via p21 expression. Nat Commun 8, 14728. [PubMed: 28317845]

Bornstein BJ, Keating SM, Jouraku A, and Hucka M (2008). LibSBML: An API library for SBML. Bioinformatics 24(6), 880–881. [PubMed: 18252737]

Chao HX, Poovey CE, Privette AA, Grant GD, Chao HY, Cook JG, and Purvis JE (2017). Orchestration of DNA damage checkpoint dynamics across the human cell cycle. Cell Syst 5, 445–459 e5. [PubMed: 29102360]

Correll M, Albers D, Franconeri S, and Gleicher M (2012). Comparing averages in time series data. Proc SIGCHI Conf Hum Factor Comput Syst.

Davis DM, and Purvis JE (2015). Computational analysis of signaling patterns in single cells. Semin Cell Dev Biol 37, 35–43. [PubMed: 25263011]

Gillespie DT (1977). Exact stochastic simulation of coupled chemical reactions. J Phys Chem 81, 2340–2361.

Gillespie DT (2007). Stochastic simulation of chemical kinetics. Annu Rev Phys Chem 58, 35–55. [PubMed: 17037977]

Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, Arkin AP, Bornstein BJ, Bray D, Cornish-Bowden A, et al. (2003). The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics 19, 524–31. [PubMed: 12611808]

Loewer A, Batchelor E, Gaglia G, and Lahav G (2010). Basal dynamics of p53 reveal transcriptionally attenuated pulses in cycling cells. Cell 142, 89–100. [PubMed: 20598361]

Maarleveld TR, Olivier BG, and Bruggeman FJ (2013). StochPy: a comprehensive, user-friendly tool for simulating stochastic biological processes. PLoS One 8, e79345. [PubMed: 24260203]

Spencer SL, Cappell SD, Tsai FC, Overton KW, Wang CL, and Meyer T (2013). The proliferation-quiescence decision is controlled by a bifurcation in CDK2 activity at mitotic exit. Cell 155, 369–83. [PubMed: 24075009]

Toettcher JE, Loewer A, Ostheimer GJ, Yaffe MB, Tidor B, and Lahav G (2009). Distinct mechanisms act in concert to mediate cell cycle arrest. Proc Natl Acad Sci USA 106, 785–90. [PubMed: 19139404]

Highlights

- Users can stack and align single-cell traces for different molecular reporters
- Computational models with adjustable parameters simulate cell cycle progression
- Virtual growth curves and flow cytometry assays predict cell cycle behaviors

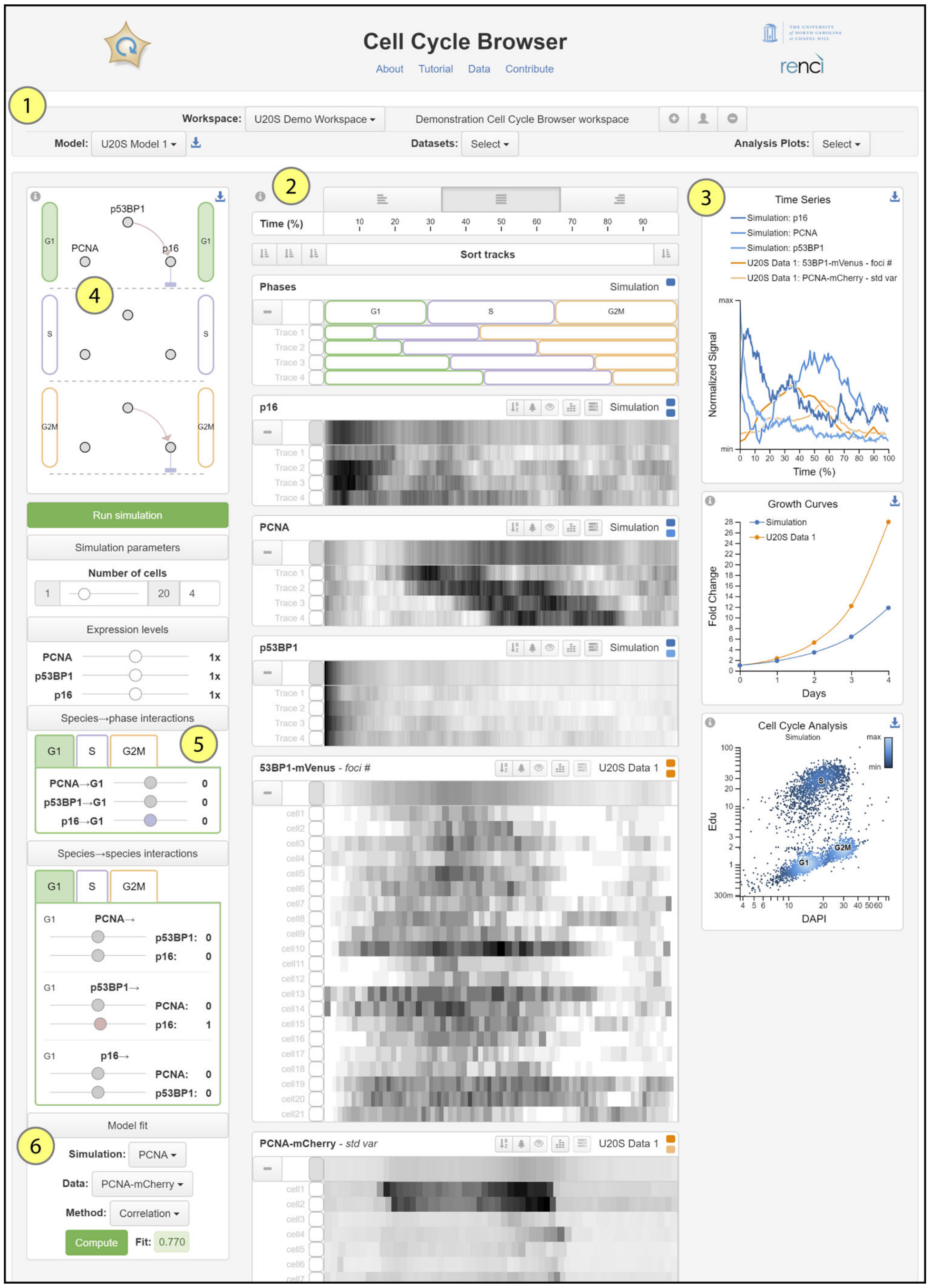

Figure 1. Cell Cycle Browser user interface

. Individual panels and functionality are described in the main text. 1. Data selection. 2. Data browser. 3. Analysis plots. 4. Model visualization. 5. Model parameter controls. 6. Model Fit panel.