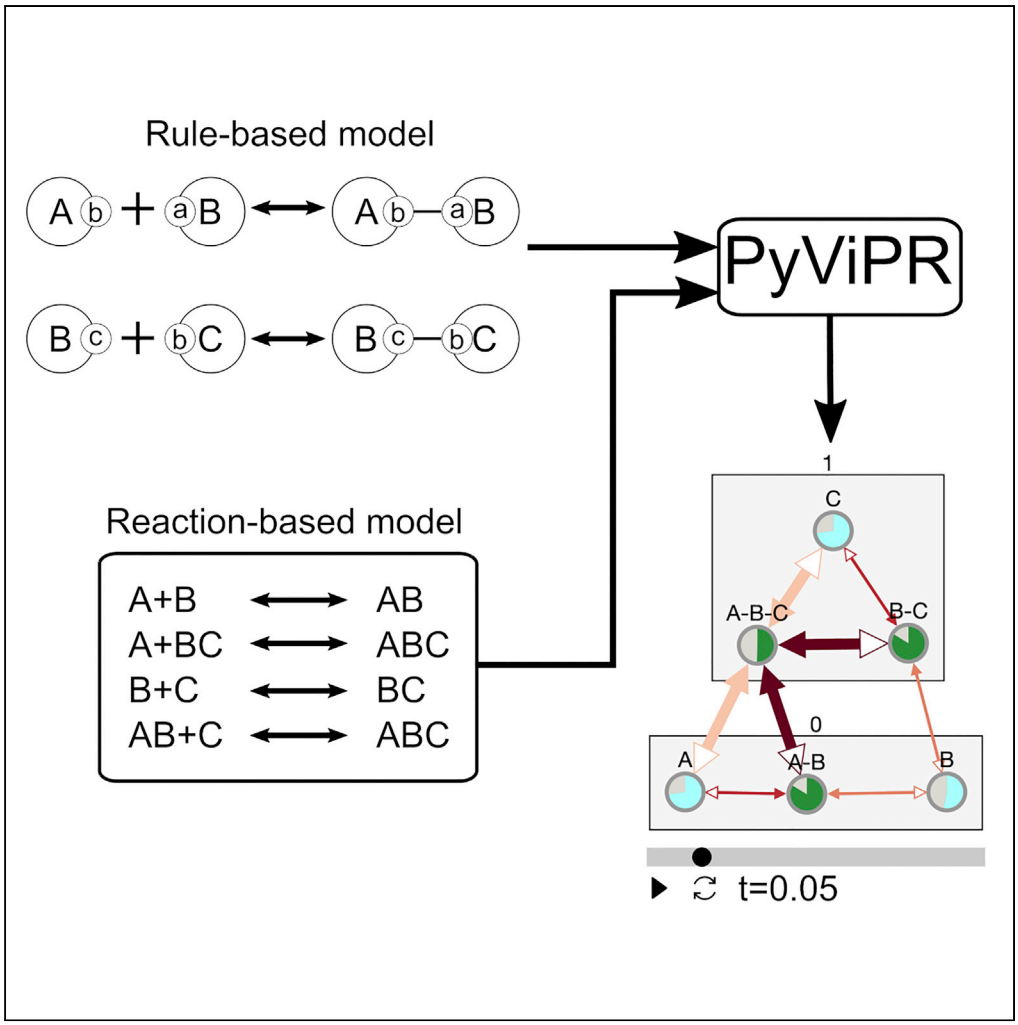
iScience

CellPress

Article

Interactive Multiresolution Visualization of Cellular Network Processes



Oscar O. Ortega, Carlos F. Lopez

c.lopez@vanderbilt.edu

HIGHLIGHTS

Static and dynamic interactive visualizations of systems biology models

Community detection algorithms are used to facilitate network exploration

Visualization embedded in Jupyter Notebooks for easy model pipeline dissemination

Support for multiple graph formats including GraphML, SBGN, and SIF

Ortega & Lopez, iScience *23*, 100748
January 24, 2020
https://doi.org/10.1016/j.isci.2019.100748

iScience

CellPress

Article

Interactive Multiresolution Visualization of Cellular Network Processes

Oscar O. Ortega[1] and Carlos F. Lopez[1,2,3,*]

SUMMARY

Visualization plays a central role in the analysis of biochemical network models to identify patterns that arise from reaction dynamics and perform model exploratory analysis. To facilitate these analyses, we developed PyViPR, a visualization tool that generates static and dynamic representations of biochemical network processes within a Python-based environment. PyViPR embeds network visualizations within Jupyter notebooks, thus enabling integration with modeling, simulation, and analysis workflows. To present the capabilities of PyViPR, we explore execution mechanisms of extrinsic apoptosis in HeLa cells. We show that community-detection algorithms identify groups of molecular species that capture key biological functions and ease exploration of the apoptosis network. We then show how different kinetic parameter sets that fit the experimental data equally well exhibit significantly different signal-execution dynamics as the system progresses toward mitochondrial outer-membrane permeabilization. Therefore, PyViPR aids the conceptual understanding of dynamic network processes and accelerates hypothesis generation for further testing and validation.

INTRODUCTION

Cellular signaling pathways are controlled by networks of biomolecular interactions that process signals from environmental cues (Lemmon and Schlessinger, 2010; Blinov et al., 2006; Sachs et al., 2005). These molecular networks give rise to nonlinear dynamic processes that are difficult to explain and predict using reductionist methods (Ahn et al., 2006). Mathematical models of cellular signaling pathways have become commonplace to gain insights and describe the molecular mechanisms that control cellular processes (Gaddy et al., 2017; Perry et al., 2019; Albeck et al., 2008). In general, these models continue to grow in size and complexity, which makes the exploration of network structure and dynamics increasingly challenging. Visualization tools present one effective way to explore network processes and acquire conceptual insights about signal-execution mechanisms. In addition, visualization tools can facilitate the detection of execution patterns and aid in hypothesis generation for experimental validation. However, most tools focus on static single-resolution network representations of models and generally lack support to visualize model dynamics. Therefore, there is an unmet need for tools that facilitate multi-resolution visualizations of model networks and simulated dynamics.

Numerous tools have been developed to visualize network representations of models that capture relationships between model components. Some examples include molecular species networks (Bergmann et al., 2017), hierarchical species networks (Paduano and Forbes, 2015), species-reactions networks (Schaff et al., 2016), contact maps (Harris et al., 2016; Boutillier et al., 2018; Cheng et al., 2014), model-defined rules (Boutillier et al., 2018; Cheng et al., 2014), and rule-based networks (Smith et al., 2012; Danos et al., 2012), among many others (Kolpakov et al., 2019; Tiger et al., 2012; Dang et al., 2015). Although these tools have been highly useful within their domains, they exhibit limitations when it comes to visualizing the structures of increasingly complex networks with an ever-larger number of nodes and edges labels. Moreover, standalone visualization tools can be difficult to incorporate into model-building and analysis workflows, further compounding the lack of reproducibility in analysis pipelines.

Identifying reactions that drive cellular processes is central to dynamic network analysis, yet it is highly challenging without visualization tools to facilitate an intuitive understanding of the signal execution mechanisms. A handful of tools to visualize dynamic network processes have been published, notably COPASI (Bergmann et al., 2017) and the Kappa Dynamic Influence Network (KDIN) (Forbes et al., 2017). COPASI uses a network in which nodes represent biochemical species and edges represent biochemical interactions. Species concentrations obtained from a simulation are encoded in the size of the box around the network nodes. Kappa employs a network in which nodes are the model rules and the edges indicate

[1]Chemical and Physical Biology Program, Vanderbilt University, Nashville, TN, USA

[2]Biochemistry Department, Vanderbilt University, Nashville, TN, USA

[3]Lead Contact

*Correspondence: c.lopez@vanderbilt.edu

https://doi.org/10.1016/j.isci.2019.100748

that the rules have common reactant or product species. KDIN quantitatively represents the temporal influence that each biochemical rule exerts on other rules. Although both tools yield useful information about dynamic network processes, information about the reactions that drive the dynamic consumption and production of different biomolecules, essential to understanding signal execution mechanisms, is not easily obtained. In addition, these tools have been developed for software-specific environments, thus limiting their use in general modeling and analysis workflows.

In this work we tackle three main visualization challenges that, we believe, will accelerate the conceptual understanding of biological network processes: (1) develop legible and comprehensible visualizations of increasingly large networks; (2) generate intuitive dynamic network visualizations of model simulations; and (3) facilitate the integration of visualizations into model building and analysis pipelines. To tackle these challenges, we developed Python Visualization of Processes and Reactions (PyViPR), a Python-based framework that provides multiple static and dynamic representations of biological processes. Importantly, PyViPR unifies tools typically used in isolation, enables access to community-detection algorithms, and encodes model simulations into node and edge attributes, thus enabling the study of network dynamics at multiple resolutions. PyViPR embeds all its visualization and analysis capabilities within Jupyter Notebooks (Kluyver et al., 2016) to facilitate reproducibility and dissemination of model analysis pipelines. PyViPR currently supports the rendering of rule-based models declared in the PySB framework (Lopez et al., 2013), BioNetGen (BNG) (Harris et al., 2016), and Kappa language (Boutillier et al., 2018), as well as models encoded in the SBML format (Hucka et al., 2003), thus providing a general tool to visualize models of biochemical network processes. In what follows we describe PyViPR's design and implementation, followed by a demonstration of key PyViPR capabilities in the exploration of cellular signal processing.

RESULTS

PyViPR Overview

PyViPR is a Python package that operates within the Jupyter notebooks environment (Kluyver et al., 2016). In this manner, PyViPR takes full advantage of a Literate Programming paradigm (Knuth, 2001), which enables the definition of both code and documentation concurrently and allows users to develop shareable workflows for model definition, visualization, and analysis. PyViPR leverages the capabilities of PySB to generate model objects, import models from BNGL and SBML formats, and provide simulation-based results for dynamic visualization. In addition, PyViPR integrates Cytoscape.js (Franz et al., 2015), a well-established JavaScript library for graph visualization, into the Python environment to interactively render static and dynamic visualization of model networks. Therefore, PyViPR merges software packages that would traditionally be used in isolation onto a common modeling environment. Further, PyViPR benefits from community-driven software development, and improvements made to any of its software dependencies are automatically accrued by the framework. PyViPR encourages community-driven collaboration through its open-source philosophy built around GitHub: https://github.com/LoLab-VU/PyViPR.

A typical PyViPR workflow comprises the following steps. First, a supported model file is passed to one of the PyViPR visualization functions. PyViPR then uses NetworkX (Hagberg et al., 2008) to convert the model components into graph nodes and edges. The user could then simplify the graph through community detection (e.g., with the Louvain algorithm [Blondel et al., 2008]) on the NetworkX graph object. The software will then create a compound node and place all the nodes from a community within it. For dynamic visualization, PyViPR maps the simulated species concentrations and reaction data to node and edge properties. The resulting NetworkX graph is transferred to cytoscape.js via a JSON dictionary and rendered real-time in a Jupyter notebook for visualization. We note that the user can interact with all graph objects in a Jupyter notebook rendering to, e.g. change the layout, groupings, or placement of a given graph.

PyViPR supports visualization of the two main approaches used to build chemical kinetics models of cellular regulatory networks. In the first approach, reaction networks are generated by enumerating all the molecular species and reactions that can occur in a cellular process. This reaction network can then be translated into a set of Ordinary Differential Equations (ODEs) or stochastic equations (Aldridge et al., 2006). In the second approach, rule-based modeling formalisms (Faeder et al., 2009; Boutillier et al., 2018; Lopez et al., 2013) are used to circumvent the need to enumerate all the species and reactions by hand. In these formalisms, species are defined as structured objects that can have binding and state sites, and reaction rules define interactions between specific domains or binding sites on a given species. Then, rule-based modeling tools automatically generate a reaction network by identifying all possible

species that have the conditions required to undergo the interaction defined in a rule. PyViPR supports visualization of both model encodings through a Tellurium (Choi et al., 2018) interface for reaction network models and a PySB (Lopez et al., 2013) interface for rule-based models.

In addition to biochemical network visualization, PyViPR supports the following graph formats widely used in the systems-biology community: GraphML, SIF, SBGN XML, Cytoscape JSON, GEXF, GML, and YAML. Additionally, rendered graphs in a Jupyter Notebook can be downloaded in the following formats: PNG, SIF, GraphML, and JSON.

Design Choices for PyViPR

Numerous approaches have been developed to visualize temporal networks. Beck et al. (Beck et al., 2017) surveyed a range of existing tools and derived a taxonomy based on temporal representation, either as an animation or as a static timeline. From this perspective, PyViPR would be classified as a hybrid visualization that uses the node-edge paradigm to visualize networks, an animation for visual representation of time, and superimposition of pie charts embedded in nodes as well as edges width and color, to represent the temporal changes in species concentration and reaction flux, respectively. PyViPR was designed with the following visualization goals:

G.1 Highlight functionally related species by grouping them in compound nodes.

G.2 Understand how a signal is executed in a biochemical network and how it depends on parameter values.

G.3 Provide easy-to-use interactive visualizations for investigating the topology and simulation results of biochemical models.

Murray et al. (Murray et al., 2017) identify a task taxonomy for biological pathways analysis across three categories: attribute, relation, and modification tasks. PyViPR specifically supports attribute tasks, to obtain information about a species node, and relationship tasks to identify types of relationships between nodes (e.g. protein binding, protein translocation), the direction of nodes interactions, and grouping relationships (e.g. model compartments, communities). With respect to the temporal features tasks described in the task taxonomy of network evolution analysis by Ahn et al. (Ahn et al., 2014), PyViPR focuses on the temporal features of aggregated events. More specifically, PyViPR aims to make it easy to observe at any point in time, the reaction rates that have a higher flux than other reactions.

To satisfy the design criteria introduced above, we made the following design choices:

DC.1 Employ node-link diagrams for all static and dynamic visualizations to show interactions between model components. We decided to use node-link diagrams because they are commonly used by biology experimentalists (Demir et al., 2010; Cerami et al., 2010) and computational modelers (Murray et al., 2017) and that would facilitate the interpretation and communication of results.

DC.2 Dynamically map simulated species concentrations and reaction rates values onto pie charts embedded within nodes and edge color and width, respectively. Our main goal is to clearly show the reactions that carry most flux and drive the behavior of the system over time. Therefore, we followed the design principles for the representation of flow quantity and direction discussed by Bernhard et al. (Jenny et al., 2018) and used color brightness to represent reaction flow quantity on edges.

DC.3 Include search mechanisms, multiple layout options, zoom and grouping functionality to organize model components, and focus on important details, thus enabling interactive exploration of complex biological networks.

Network Creation from Multiple Model Components

PyViPR supports visualization of multiple model components, including molecular species, reactions, rules, compartments, macro functions (Lopez et al., 2013), and modules comprising independent model elements (Lopez et al., 2013). These components are depicted by either *simple nodes*, which are fundamental units in a graph (Figure 1A), or *compound nodes*, which can contain children nodes and are used to group simple nodes with shared attributes or through user-defined groupings. Interactions between

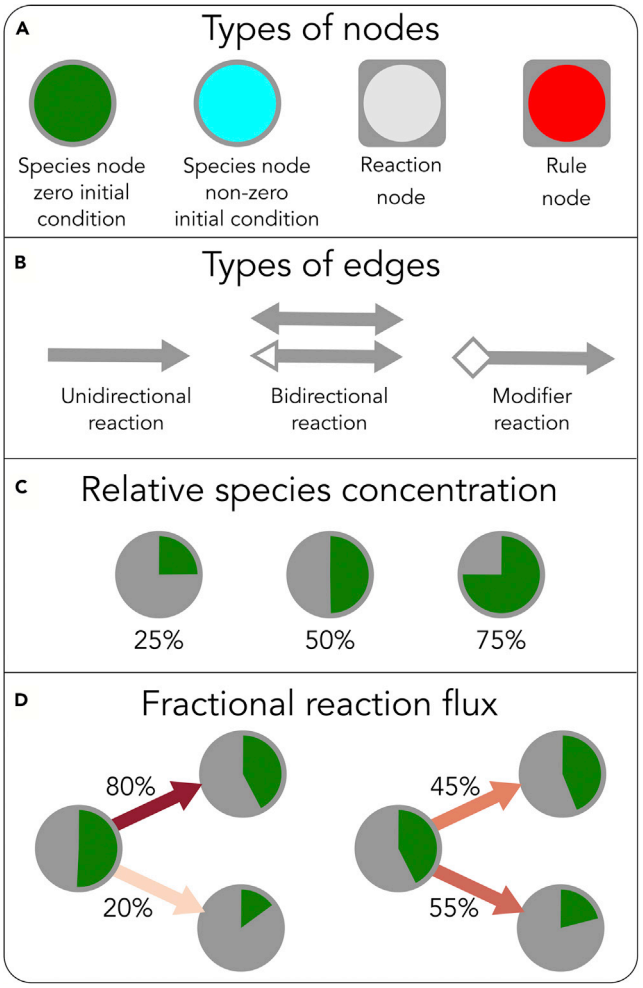


Figure 1. PyViPR Visual Encodings

(A) Node types used for visualizations as labeled.

(B) Edge types used for interactions: *unidirectional interactions* (left) are depicted with a unidirectional arrow and represent irreversible biochemical reactions. *Bidirectional interactions* (middle) are depicted with bidirectional arrows and represent reversible reactions. Arrows fill state indicate directionality from reactant (hollow) to product (solid) species. Solid bidirectional arrows represent bidirectional interactions lacking directionality information. *Modifier reaction* (right) are depicted with an arrow tail shaped with a hollow diamond and a solid arrow head and represent reactions where the species is both a reactant and a product of the reaction.

(C) Pie charts embedded within nodes indicate the concentration of a species relative to its maximum value in the simulation.

(D) Color shade of arrows indicate the fractional reaction flux for interactions.

these different model components correspond to unidirectional or bidirectional reactions and are represented by arrows (Figure 1B).

To create a bipartite network, PyViPR first obtains the list of species and rules/reactions from a model and adds them as nodes to the network. Then, PyViPR uses edges to connect species nodes with their respective rule/reaction node. To reduce the network resolution a bipartite graph can be projected onto a unipartite graph that contains only the species or rules/reactions nodes (see Figure S1A—Unipartite graph). This unipartite species graph can then be organized by grouping the species nodes using the biological compartments on which they are located (See Figure S1A—Compound graph). Similarly, a unipartite rules graph can be grouped by the macro functions used to create them or the model modules where they are defined. This allows users to interactively explore and revise the model network topology at different

resolutions. For a complete list of the different model components that can be visualized in a network see Figure S2.

A key feature in PyViPR is the use of community detection algorithms to automatically cluster nodes and thereby simplify network complexity. For example, the Louvain method detects communities by optimizing the graph modularity. In this method, optimization is achieved by first iterating over all nodes and assigning each node to a community that results in the greatest local modularity increase, then each small community is grouped into one node and the first step is repeated until no modularity increase can occur (Blondel et al., 2008). As a result, the Louvain algorithm finds groups of highly connected nodes that could have similar biological functions or represent molecular-complex formation processes (Fortunato, 2010) (design goal 1). Other community detection algorithms based on label propagation (Raghavan et al., 2007; Cordasco and Gargano, 2010), fluid communities (Parés et al., 2018), and centrality (Girvan and Newman, 2002) methods are also available in PyViPR. Alternatively, users can also manually define clusters of nodes interactively for a "human in the loop" type optimization (Daschinger et al., 2017; Holzinger, 2016). Taking advantage of the PySB interface to BioNetGen, we also incorporated (1) compact rule visualization, (2) atom-rule graph, and (3) tunable compression pipeline as implemented by Sekar et al. (Sekar et al., 2017) into the PyViPR workflow to enable a more thorough and complete visualization of large rule-based models.

Dynamic Visualization in PyViPR

PyViPR supports dynamic visualization of deterministic and stochastic model simulations (See Figure S1D). This visualization mode uses a unipartite network (Design Choice DC.1) in which nodes represent model species and edges represent reactions between the species. Species concentrations and reaction rates are encoded into the properties of nodes and edges, respectively (Design Choice DC.2).

To represent temporal concentration changes during a simulation, we embedded pie charts within the graph species nodes. Pie chart slices within each node depict the species concentration relative to the maximum amount attained throughout the simulation. Pie chart slices are updated at each time point during animation (Figure 1C). Absolute species concentrations at a given time point are also accessible as tooltips through a click-hold gesture on a species node.

PyViPR aims to highlight reactions with high rates of consumption or production, as these can drive complex network processes (Design Goal 2). To attain this goal, simulated reaction rates are encoded on both the color shade and the thickness of arrows that connect interacting species. For each species PyViPR obtains its related reactions and then calculates the fractional flux of each reaction using a normalization function:

$$f_{i,c}(t) = \frac{r_{i,c}(t)}{\sum_{j=1}^{n} r_{j,c}(t)}$$

where $r_{i,c}$ is the reaction rate value at a specific time point, n is the number of reactions, and the sub-index c indicates the type of reaction (consumption or production). Fractional fluxes are then linearly mapped to a color shade ranging from low (light shade) to high (dark shade) flux representations (Figure 1D). In addition, reaction rate values, relative to their maximum value throughout the simulation, are represented by edge thickness. Absolute reaction rate values for each interaction and at any given time point are also accessible as tooltips using the click-hold gesture.

Exploration of a Biological Process with PyViPR: Apoptosis Execution

To illustrate the visualization capabilities of PyViPR, we use the Extrinsic Apoptosis Reaction Model (EARM v2.0) (Lopez et al., 2013) to perform an exploratory analysis of the receptor-mediated apoptosis signaling network. Briefly, EARM v2.0 describes the biochemical interactions from an initial death ligand cue to a cleaved PARP response. Initiator caspases trigger interactions among the Bcl-2 family of proteins that lead to mitochondrial outer membrane permeabilization (MOMP). MOMP, in turn, propagates the signal to effector caspase activation and PARP cleavage. EARM is a sizable model that comprises 74 molecular species, 127 parameters, 62 rules, and 100 reactions. We explored the EARM network using the following steps: (1) visualization of the apoptosis species-rules bipartite network; (2) application of the Louvain community detection algorithm to functionally cluster species nodes; (3) study of the simulation dynamics

at a coarse-grained community level; and (4) identification of molecular targets that modulate model behavior.

Multiresolution Visualization and Exploration of EARM

We wanted to study the architecture of the network defined in EARM to find insights about molecular organization and function in apoptosis execution. We first visualized a species-rules bipartite network (Figure 2, upper panel). However, this network is difficult to explore, as no discernible structures are readily apparent. We then projected the species-rules bipartite graph onto a species unipartite graph and clustered highly connected nodes using the Louvain algorithm (Figure 2, middle panel). These communities can also be further collapsed to obtain the EARM communities graph, a coarse-grained representation of the apoptosis pathway (Figure 2, lower panel).

The Louvain community detection algorithm identified nine communities, numbered 0–8, which is summarized in Table 1 (see Figure S3). Briefly, these communities capture biologically relevant and functional processes throughout the apoptosis pathway. Community 1, describing Caspase 8 activation and Bid truncation, is linked with Communities 3 and 4, the starting points for type I and type II cellular apoptosis, respectively (Özören and El-Deiry, 2002). Interestingly, Mcl-1, a potent apoptosis inhibitor, was placed in a separate community from all the other Bcl-2 inhibitors, highlighting its unique inhibitory interactions that have been well documented (Yang-Yen, 2006). Community 4 is also connected to Communities 5 and 8 that correspond to Bak and Bax activation, polymerization, and pore formation, respectively. These communities capture mitochondrial regulation events that lead to eventual MOMP formation in type II apoptosis execution (Kale et al., 2017; YIN, 2000). These MOMP-related communities are connected to Communities 2 and 7, which correspond to MOMP-driven release of cytochrome c and Smac from the mitochondria. Finally, these communities connect to Community 3, which corresponds to the activation of executioner Caspase 3 (C3) and subsequent PARP cleavage, which signals that the cell has executed apoptosis. As shown, C3 can be directly activated by Caspase 8 (C8) (type I) or by the apoptosome formed after cytochrome c is released via MOMP (type II).

The Louvain algorithm also led to some interesting observations regarding molecular interactions. For example, Caspase 3, the effector caspase, is the species with the highest within-community node degree, indicating that it is a highly regulated protein in apoptosis execution. Also, mBid has the highest number of interactions across communities, indicating that it plays a key regulatory role in apoptosis execution.

Taken together, we find that Louvain community detection could be used as an interactive "coarse-graining" methodology to automatically group biochemical interactions, simplify mechanism exploration, and identify important proteins within a biochemical network.

Parameter Sets Fit Experimental Data but Yield Different Network Dynamics

To demonstrate the advantages of dynamic visualization, we calibrated EARM to previously published time course experimental data (Spencer et al., 2009) using the Particle Swarm Optimization (PSO) algorithm (Kennedy and Eberhart, 1995). Ten thousand PSO runs were carried out, which resulted in 6,572 parameter sets with an error $\leq$ 2.8 (See Methods section for details). It is well established that multiple parameter sets can fit experimental data equally well, due to parameter unidentifiability and model sloppiness (Gutenkunst et al., 2007). To explore the mechanistic implications of different parameter sets on EARM execution, we compared the dynamics generated by two different parameter sets, labeled Parameter Set 1 (PS1) and Parameter Set 2 (PS2) (Table S1), as described below.

We hypothesized that these two parameter sets with different kinetic parameter values would yield distinct signal mechanisms. Thus, we first asked whether a trajectory plot of tBid dynamics could yield useful mechanistic information about apoptosis execution with different parameter sets. As shown in Figure 3A, both parameter sets generated tBid trajectories that were essentially indistinguishable, yielding no mechanistic information from the two distinct parameter sets. We then employed the EARM communities graph to compare the global dynamic signal execution for both parameter sets. Figure 3B shows three time points in signal execution for PS1 (upper panel) and PS2 (lower panel). As shown, there is little activity between communities in both parameter sets in the early time points (Figure 3B left). However, for PS1 at t = 4040s we observe that Community 1, which regulates C8 activation, exhibits increased flux toward Community 3, which controls C3 activation. This indicates that C3 is being activated by C8. Despite the activation

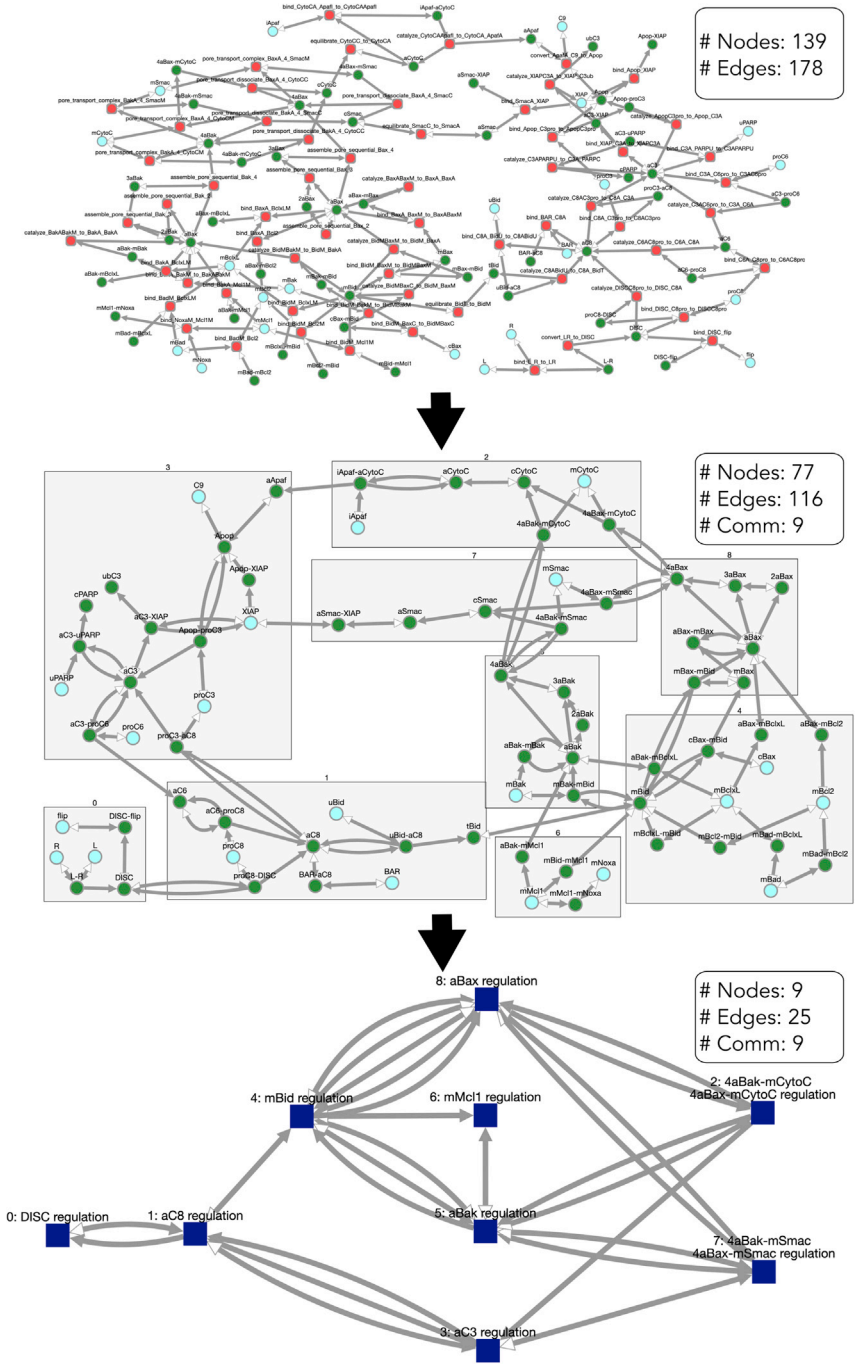


Figure 2. Multiresolution Visualization of a Reaction Network

Upper panel: EARM species rules bipartite graph. Green nodes represent molecular species with initial condition set to zero, cyan nodes are species with nonzero initial conditions, and red nodes represent rules defined in the model. *Middle panel*: EARM species graph obtained from projecting the bipartite graph into a unipartite graph. Densely connected nodes have been automatically grouped into communities detected with the Louvain algorithm. *Lower panel*: EARM communities graph obtained by collapsing each communities into a single node. Community node names are assigned by the species with the highest number of interactions within its community. All edges correspond to interactions between species nodes as specified in the EARM model.

Community Number	Apoptosis Subprocess	References
0	Ligand-receptor interactions that lead to the DISC formation and regulation by Flip	Pennarun et al., 2010
1	Initiator Caspase 8 activation by DISC and Caspase 6 and subsequent truncation of Bid by activated Caspase 8	Kantari and Walczak, 2011
2	Release of cytochrome C through mitochondrial Bax and Bak pores	Garrido et al., 2006
3	Activation and regulation of effector Caspase 3, formation of apoptosomes, and cleavage of PARP	Zou et al., 2003
4	Bcl-2 family of interactions responsible for translocation of Bax to the mitochondria by mitochondrial Bid (mBid) and inhibition of Bax, Bak, and mBid by BclxL and Bcl2	Kale et al., 2017; YIN, 2000
5	Formation of mitochondrial Bax pores	Annis et al., 2005; Westphal et al., 2014
6	Mcl1 inhibition of pro-apoptotic proteins	Yang-Yen, 2006
7	Release of Smac through mitochondrial Bax and Bak pores and its subsequent inhibition by XIAP	Deng et al., 2002
8	Formation of mitochondrial Bak pores	Westphal et al., 2014; Dewson et al., 2009

Table 1. Summary of the Biological Functions Enclosed in Each Community

of the effector Caspase, apoptosis does not take place because the antiapoptotic XIAP inhibits active C3 activity. Community 4, which regulates mBid, also exhibits increased signal flux toward the Community 8 (active Bax regulation), indicating that the pores formed in the mitochondria are dominated by Bax oligomerization.

PS2 also exhibited increased signal flux between communities but with different interaction patterns compared with those seen in PS1. Specifically, it exhibited significant flux between Community 4 and Community 6 (mMcl1 regulation) at t = 7474s, suggesting that mBid was being inhibited by mMcl1. Also, there was significant signal flux from Community 2, which regulates cytochrome c release from the mitochondria, toward Community 3, indicating that pores were already formed in the mitochondria and cytochrome c was being released to aid with the formation of the apoptosome. Therefore, dynamic visualization of signal flow across communities confirms our hypothesis about signal execution and demonstrates the usefulness of PyViPR to explore the complex dynamics that occur in biochemical processes.

To further explore the effects of kinetic parameters in model behavior, we focused on local signal flow through mBid and its interactions, as they are tightly linked to MOMP and cellular time-to-death (Spencer et al., 2009). As shown in Figure 3C, for PS1 we observed that most of mBid was used to transport cytosolic Bax to the mitochondrial outer membrane (MOM), whereas no activation of Bak occurred, suggesting that pore activity in the MOM was primarily due to Bax (see Video S1). We, therefore, hypothesized that the model with PS1 depends on Bax for apoptosis execution. In contrast, for PS2, we observed that mBid activity was primarily inhibited by the anti-apoptotic Mcl1 (see Video S2). We thus hypothesized that under PS2 an MCL-1 knockdown would free mBid to activate Bax and Bak and more rapidly commit cells to apoptosis. We tested both hypothesis derived from our visualization-based analysis using *in silico* experiments. First, we knocked out Bax and simulated EARM with PS1 (Figure 4). We found that knocking out Bax protected cells from apoptosis induction with TRAIL, confirming that Bax plays an important role in apoptosis regulation. We then knocked out Mcl1 and simulated EARM with PS2. We found that the time-to-death was reduced by 22.6%, corroborating that Mcl1 inhibition delayed apoptosis. As a control, we knocked out Mcl1 for PS1 and Bax for PS2 and found that the dynamics of cPARP were not considerably affected.

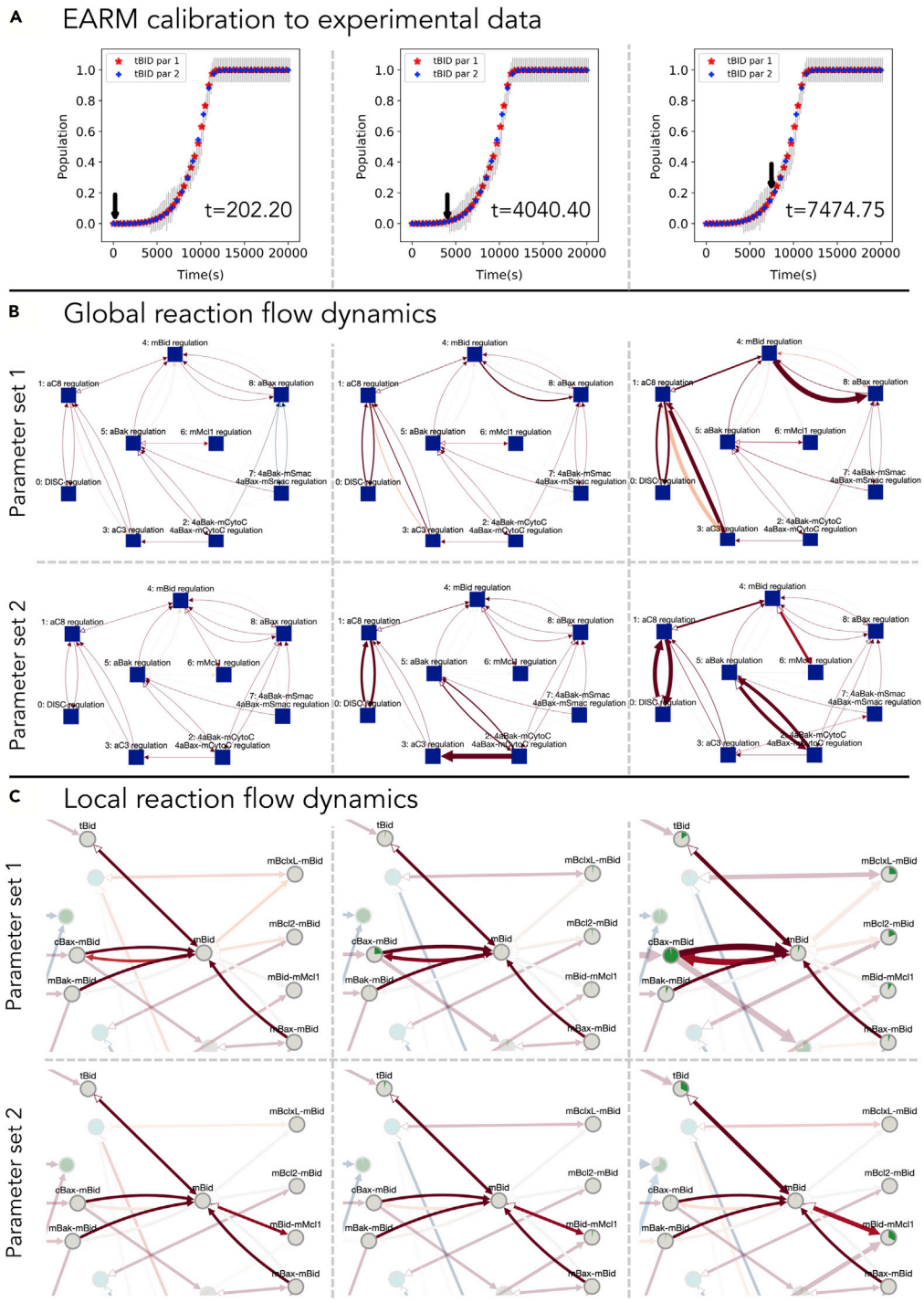


Figure 3. Dynamic Visualization of EARM at Different Resolutions

Panel (A) includes three plots of the simulated tBid and the experimental data used for calibration. Gray lines correspond to the experimental data with error bars indicating the standard deviation. Arrows indicate the concentration level at the corresponding time point. The time points shown here are the same ones used to obtain snapshots of the dynamic visualization in the following panels. Panel (B) shows, for PS1 and PS2, the global reaction flow dynamics between the communities detected with the Louvain algorithm. Panel (C) shows, for PS1 and PS2, the temporal changes in the strength of the interactions between mitochondrial Bid and the anti-apoptotic and pro-apoptotic proteins. Pie chart slices within the nodes show the concentration of a species relative to the maximum amount of the concentration attained across all time points of the simulation. Edges color shade and thickness represent the fractional flux and relative reaction value, respectively.

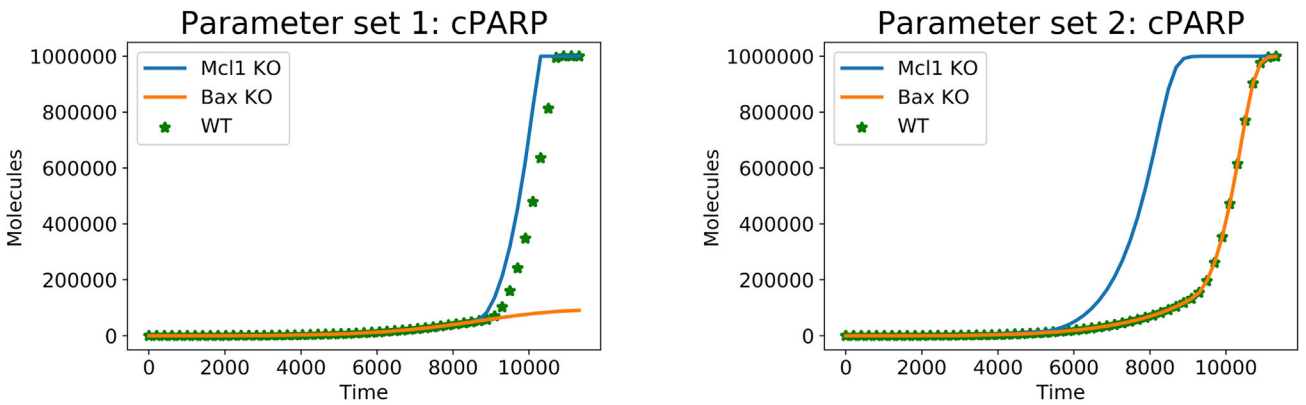


Figure 4. *In Silico* Knock outs of Bcl-2 Proteins Modulates PARP Cleavage

EARM was run using Parameter Set 1 (left) and Parameter Set 2 (right) and with knockout (KO) of either Bax or Mcl-1 as labeled. Bax KO has a significant effect on the dynamics of PARP cleavage (yellow line) for Parameter Set 1 but almost no noticeable effect for Parameter Set 2 (right). In contrast, Mcl-1 KO has almost no effect for Parameter Set 1 (left) but a significant effect in Parameter Set 2 (right).

Taken together, our results demonstrate that despite multiple parameters fitting the data equally well, apoptosis is executed differently for each parameter set. Our observations align with experimental results that show cellular dependence on Bcl-2 regulators for apoptosis execution (Deng et al., 2002; Zhou et al., 1997). Importantly, visualization of the dynamic process enabled us to identify key reactions under different parameter sets and generate testable hypotheses to better understand the execution mechanism.

DISCUSSION

In this paper, we presented PyViPR, a tool to visualize the structure and dynamics of biochemical network models. PyViPR enables a straightforward workflow of model creation, analysis, visualization, and hypothesis generation. Additionally, PyViPR integrates community detection algorithms to organize the nodes of biochemical networks and ease the exploration of complex networks. Lastly, PyViPR provides an interface for intuitive dynamic visualization that facilitates the observation of signal flow across biochemical models.

Multiple tools exist for static visualization of biological networks. Some of the tools used to visualize reaction-based models include Dynetica (Eidum et al., 2014), COPASI (Bergmann et al., 2017), CySBML (König et al., 2012), and Omix (Droste et al., 2011). Although these tools provide useful visualizations of biochemical models, they are implemented as Graphical User Interfaces, which can hinder the creations of pipelines for model creation, visualization, and analysis. Also, these tools can become difficult to use as network complexity increases. PyViPR aims to address these issues by enabling access to community detection algorithms for network simplification and facilitating the model definition, visualization, and analysis pipelines in a single Jupyter Notebook environment.

Various tools for visualization of rule-based models have also been published. These include Simmune (Cheng et al., 2014), BioNetGen (Smith et al., 2012; Sekar et al., 2017), rxncon (Tiger et al., 2012), Virtual Cell (Vasilescu et al., 2018), and Kappa (Boutillier et al., 2018). All of these tools take advantage of the structured definition of molecules and rules to generate intelligible visualizations of large models. PyViPR does not use these structured definitions and instead uses the rule-based modeling framework to obtain the set of reactions from a given model. This set of reactions is often larger than the number of rules, thus limiting the size of models that can be intelligibly visualized with PyViPR. To address this potential shortcoming, we leveraged the flexibility of a Python-based environment and provided an interface to BioNetGen's atom-rules graph algorithm.

Visualization tools to explore the dynamics of temporal network processes can be classified into three groups based on the components animated: (1) Species nodes animation, where the simulated species concentration is mapped onto the size/color of nodes (e.g. COPASI [Bergmann et al., 2017], Narrator [Xia et al., 2011], CytoModeler [Mandel et al., 2007]); (2) species nodes and edge animation, where the simulated species concentration is mapped onto the size/color of nodes, whereas reaction rate values are encoded into the edge thickness (e.g. DBSolve [Gizzatkulov et al., 2010]); and (3) rules nodes and edges

animation (e.g. DIN-Viz [Forbes et al., 2017]), where the number of hits of a rule is mapped into the node size, and the influence of one rule on another is encoded into the edge width. Similar to the first two groups of dynamic visualization approaches, PyViPR maps the species concentrations into nodes. The main difference, however, is that PyViPR encodes the reaction rates into edges width and colors in a more insightful way as it highlights the edges that carry most of the signal flow. Additionally, PyViPR is better suited for dynamic visualization of large networks, as it can use community detection algorithms to cluster nodes and then animate the coarse-grained network with the simulation results. Lastly, it is difficult to compare PyViPR with the third group of visualization tools, as PyViPR uses a species graph, whereas the latter uses a rules graph to encode the simulation results. However, one advantage that PyViPR has is that the visualization can be easily communicated to non-modeling scientists, as it only requires knowledge about the biological network being studied.

We believe that PyViPR could be incorporated into existing modeling and simulation workflows provided by Python-based tools such as Tellurium notebooks (Medley et al., 2018) and PySCeS (Olivier et al., 2004). In the future, we plan to incorporate community-detection algorithms that consider the weight of the edges for the clustering of nodes. Additionally, we plan to improve the synchronization from the JavaScript frontend to the Python backend to enable users to interactively modify model parameters and components.

All the model exploratory analyses, which include model calibration, visualization, hypothesis exploration, and testing, were performed in Jupyter Notebooks. These shareable and reusable notebooks contain all the source code and markup text that explains the rationale for each step in the analysis. We believe that access to these resources will promote reproducibility and transparency by enabling other researchers to rerun or expand the presented model analysis. We invite the community to contribute to open-source tools such as PyViPR to improve model analysis and visualization (see Supplement Information Section and https://mybinder.org/v2/gh/LoLab-VU/PyViPR/master).

Limitations of the Study

Although PyViPR can visualize a broad range of systems biology models, it is not a panacea for model visualization. Specifically, PyViPR has limitations to generate intelligible networks of rule-based models with rules that generate a few hundreds of molecular reactions. This limitation emerges because PyViPR visualizations are created from the molecular reactions, which are typically more numerous than model rules, instead of the monomers and rules encoded in a model. In this case, specialized visualization tools such as atom-rules (Sekar et al., 2017), rxncon (Tiger et al., 2012), and Kappa (Boutillier et al., 2018) could be better suited to obtain intelligible visualizations of rule-based models.

METHODS

All methods can be found in the accompanying Transparent Methods supplemental file.

DATA AND CODE AVAILABILITY

The python package PyViPR is an open-source project under the MIT License. Stable releases of PyViPR are available on PyPi, and the latest unreleased version can be downloaded from GitHub (https://github.com/LoLab-VU/PyViPR). The documentation with examples and description of the available functions is available at https://PyViPR.readthedocs.io. A Jupyter notebook with the code to reproduce all the figures included in the manuscript can be found in binder https://mybinder.org/v2/gh/LoLab-VU/PyViPR/master.

SUPPLEMENTAL INFORMATION

Supplemental Information can be found online at https://doi.org/10.1016/j.isci.2019.100748.

ACKNOWLEDGMENTS

We thank Blake Wilson, Leonard Harris, and Alexander Lubbock for their useful insights throughout the development of PyViPR and for useful feedback in writing this manuscript. This work was supported by National Science Foundation (NSF) award MCB 1411482 to CFL, National Institutes of Health (NIH) award 1U01CA215845 to CFL, the Vanderbilt International Students Program to OOO, and the Vanderbilt Pearson Graduate Fellowship to OOO.

AUTHOR CONTRIBUTIONS

Conceptualization, O.O.O and C.F.L.; Software, O.O.O.; Writing—Original Draft, O.O.O.; Writing—Review & Editing, C.F.L. and O.O.O.; Supervision, C.F.L.; Funding Acquisition, C.F.L.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: June 3, 2019
Revised: November 8, 2019
Accepted: November 25, 2019
Published: January 24, 2020

REFERENCES

Ahn, A.C., Tewari, M., Poon, C.-S., and Phillips, R.S. (2006). The limits of reductionism in medicine: could systems biology offer an alternative? PLoS Med. *3*, e208.

Ahn, J.W., Plaisant, C., and Shneiderman, B. (2014). A task taxonomy for network evolution analysis. IEEE Trans. Vis. Comput. Graph. *20*, 365–376.

Albeck, J.G., Burke, J.M., Spencer, S.L., Lauffenburger, D.A., and Sorger, P.K. (2008). Modeling a snap-action, variable-delay switch controlling extrinsic cell death. PLoS Biol. *6*, e299.

Aldridge, B.B., Burke, J.M., Lauffenburger, D.A., and Sorger, P.K. (2006). Physicochemical modelling of cell signalling pathways. Nat. Cell Biol. *8*, 1195–1203.

Annis, M.G., Soucie, E.L., Dlugosz, P.J., Cruz-Aguado, J.A., Penn, L.Z., Leber, B., and Andrews, D.W. (2005). Bax forms multispanning monomers that oligomerize to permeabilize membranes during apoptosis. EMBO J. *24*, 2096–2103.

Beck, F., Burch, M., Diehl, S., and Weiskopf, D. (2017). A taxonomy and survey of dynamic graph visualization. Comput. Graph. Forum *36*, 133–159.

Bergmann, F.T., Hoops, S., Klahn, B., Kummer, U., Mendes, P., Pahle, J., and Sahle, S. (2017). COPASI and its applications in biotechnology. J. Biotechnol. *261*, 215–220.

Blinov, M.L., Faeder, J.R., Goldstein, B., and Hlavacek, W.S. (2006). A network model of early events in epidermal growth factor receptor signaling that accounts for combinatorial complexity. BioSystems *83*, 136–151.

Blondel, V.D., Guillaume, J.L., Lambiotte, R., and Lefebvre, E. (2008). Fast unfolding of communities in large networks. J. Stat. Mech. Theor. Exp. *2008*, 1–12.

Boutillier, P., Maasha, M., Li, X., Medina-Abarca, H.F., Krivine, J., Feret, J., Cristescu, I., Forbes, A.G., and Fontana, W. (2018). The Kappa platform for rule-based modeling. Bioinformatics *34*, i583–i592.

Cerami, E.G., Gross, B.E., Demir, E., Rodchenkov, I., Babur, Ö., Anwar, N., Schultz, N., Bader, G.D., and Sander, C. (2010). Pathway Commons, a web resource for biological pathway data. Nucleic Acids Res. *39*, D685–D690.

Cheng, H.C., Angermann, B.R., Zhang, F., and Meier-Schellersheim, M. (2014). NetworkViewer: visualizing biochemical reaction networks with embedded rendering of molecular interaction rules. BMC Syst. Biol. *8*, 1–16.

Choi, K., Medley, J.K., König, M., Stocking, K., Smith, L., Gu, S., and Sauro, H.M. (2018). Tellurium: an extensible python-based modeling environment for systems and synthetic biology. Biosystems *171*, 74–79.

Cordasco, G. and Gargano, L.. (2010). Community detection via semi-synchronous label propagation algorithms. 2010 IEEE International Workshop on: Business Applications of Social Network Analysis (BASNA), pp. 1–8.

Dang, T.N., Murray, P., Aurisano, J., and Forbes, A.G. (2015). Reactionflow: an interactive visualization tool for causality analysis in biological pathways. BMC Proc. *9*, S6.

Danos, V., Feret, J., Fontana, W., Harmer, R., Hayman, J., Krivine, J., Thompson-Walsh, C., and Winskel, G.. (2012). Graphs, Rewriting and Pathway Reconstruction for Rule-Based Models. FSTTCS 2012-IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, 18, 276–288.

Daschinger, M., Knote, A., Green, R., and Von Mammen, S.. (2017). A human-in-the-loop environment for developmental biology. The 2018 Conference on Artificial Life: A Hybrid of the European Conference on Artificial Life (ECAL) and the International Conference on the Synthesis and Simulation of Living Systems (ALIFE), pp. 475–482.

Demir, E., Cary, M.P., Paley, S., Fukuda, K., Lemer, C., Vastrik, I., Wu, G., D'Eustachio, P., Schaefer, C., Luciano, J., et al. (2010). The BioPAX community standard for pathway data sharing. Nat. Biotechnol. *28*, 935–942.

Deng, Y., Lin, Y., and Wu, X. (2002). TRAIL-induced apoptosis requires Bax-dependent mitochondrial release of Smac/DIABLO. Genes Dev. *16*, 33–45.

Dewson, G., Kratina, T., Czabotar, P., Day, C.L., Adams, J.M., and Kluck, R.M. (2009). Bak activation for apoptosis involves oligomerization of dimers via their α6 helices. Mol. Cell *36*, 696–703.

Droste, P., Miebach, S., Niedenfuhr, S., Wiechert, W., and Noh, K. (2011). Visualizing multi-omics data in metabolic networks with the software Omix—a case study. Biosystems *105*, 154–161.

Eidum, D., Asthana, K., Unni, S., Deng, M., and You, L. (2014). Construction, visualization, and analysis of biological network models in dynetica. Quantitative Biol. *2*, 142–150.

Faeder, J.R., Blinov, M.L., and Hlavacek, W.S. (2009). Rule-based modeling of biochemical systems with BioNetGen (Humana Press), pp. 113–167.

Forbes, A.G., Burks, A., Lee, K., Li, X., Boutillier, P., Krivine, J., and Fontana, W. (2017). Dynamic influence networks for rule-based models. IEEE. Trans. Vis. Comput. Graph. *24*, 184–194.

Fortunato, S. (2010). Community detection in graphs. Phys. Rep. *486*, 75–174.

Franz, M., Lopes, C.T., Huck, G., Dong, Y., Sumer, O., and Bader, G.D. (2015). Cytoscape.js: a graph theory library for visualisation and analysis. Bioinformatics *32*, 309–311.

Gaddy, T.D., Wu, Q., Arnheim, A.D., and Finley, S.D. (2017). Mechanistic modeling quantifies the influence of tumor growth kinetics on the response to anti-angiogenic treatment. PLoS Comput. Biol. *13*, 1–23.

Garrido, C., Galluzzi, L., Brunet, M., Puig, P.E., Didelot, C., and Kroemer, G. (2006). Mechanisms of cytochrome c release from mitochondria. Cell Death Differ. *13*, 1423–1433.

Girvan, M., and Newman, M.E.J. (2002). Community structure in social and biological networks. Proc. Natl. Acad. Sci. U S A *99*, 7821–7826.

Gizzatkulov, N.M., Goryanin, I.I., Metelkin, E.A., Mogilevskaya, E.A., Peskov, K.V., and Demin, O.V. (2010). DBSolve Optimum: a software package for kinetic modeling which allows dynamic visualization of simulation results. BMC Syst. Biol. *4*, 109.

Gutenkunst, R.N., Waterfall, J.J., Casey, F.P., Brown, K.S., Myers, C.R., and Sethna, J.P. (2007). Universally sloppy parameter sensitivities in systems biology models. PLoS Comput. Biol. *3*, 1871–1878.

Hagberg, A.A., Schult, D.A., and Swart, P.J.. (2008). Exploring network structure, dynamics, and function using NetworkX. Proceedings of the

7th Python in Science Conference (SciPy), pp. 11–15.

Harris, L.A., Hogg, J.S., Tapia, J.J., Sekar, J.A., Gupta, S., Korsunsky, I., Arora, A., Barua, D., Sheehan, R.P., and Faeder, J.R. (2016). BioNetGen 2.2: advances in rule-based modeling. Bioinformatics *32*, 3366–3368.

Holzinger, A. (2016). Interactive machine learning for health informatics: when do we need the human-in-the-loop? Brain Inform. *3*, 119–131.

Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., Arkin, A.P., Bornstein, B.J., 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*, 524–531.

Jenny, B., Stephen, D.M., Muehlenhaus, I., Marston, B.E., Sharma, R., Zhang, E., and Jenny, H. (2018). Design principles for origin-destination flow maps. Cartography Geogr. Inf. Sci. *45*, 62–75.

Kale, J., Osterlund, E.J., and Andrews, D.W. (2017). BCL-2 family proteins: changing partners in the dance towards death. Cell Death Differ. *25*, 65–80.

Kantari, C., and Walczak, H. (2011). Caspase-8 and Bid: caught in the act between death receptors and mitochondria. Biochim. Biophys. Acta *1813*, 558–563.

Kennedy, J. and Eberhart, R.. (1995). Particle swarm optimization. Proceedings of ICNN'95-International Conference on Neural Networks, 4, 1942–1948.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B.E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J.B., Grout, J., Corlay, S., et al. (2016). Jupyter notebooks - a publishing format for reproducible computational workflows. In Positioning and Power in Academic Publishing: Players, Agents and Agendas, F. Loizides and B. Schmidt, eds. (IOS Press), pp. 87–90.

Knuth, D.E. (2001). Literate programming. Comput. J. *27*, 97–111.

Kolpakov, F., Akberdin, I., Kashapov, T., Kiselev, I., Kolmykov, S., Kondrakhin, Y., Kutumova, E., Mandrik, N., Pintus, S., Ryabova, A., et al. (2019). BioUML: an integrated environment for systems biology and collaborative analysis of biomedical data. Nucleic Acids Res. *47*, W225–W233.

König, M., Dräger, A., and Holzhütter, H.-G. (2012). CySBML: a cytoscape plugin for SBML. Bioinformatics *28*, 2402–2403.

Lemmon, M.A., and Schlessinger, J. (2010). Cell signaling by receptor tyrosine kinases. Cell *141*, 1117–1134.

Lopez, C.F., Muhlich, J.L., Bachman, J.A., and Sorger, P.K. (2013). Programming biological models in Python using PySB. Mol. Syst. Biol. *9*, 1–19.

Mandel, J.J., Fuß, H., Palfreyman, N.M., and Dubitzky, W. (2007). Modeling biochemical transformation processes and information processing with Narrator. BMC Bioinformatics *8*, 103.

Medley, J.K., Choi, K., König, M., Smith, L., Gu, S., Hellerstein, J., Sealfon, S.C., and Sauro, H.M. (2018). Tellurium notebooks-an environment for reproducible dynamical modeling in systems biology. PLoS Comput. Biol. *14*, 1–24.

Murray, P., McGee, F., and Forbes, A.G. (2017). A taxonomy of visualization tasks for the analysis of biological pathway data. BMC Bioinformatics *18*, 21.

Olivier, B.G., Rohwer, J.M., and Hofmeyr, J.-H.S. (2004). Modelling cellular systems with PySCeS. Bioinformatics *21*, 560–561.

Özören, N., and El-Deiry, W.S. (2002). Defining characteristics of types I and II apoptotic cells in response to TRAIL. Neoplasia *4*, 551–557.

Paduano, F., and Forbes, A.G. (2015). Extended LineSets: a visualization technique for the interactive inspection of biological pathways. BMC Proc. *9*, S4.

Parés, F., Gasulla, D.G., Vilalta, A., Moreno, J., Ayguadé, E., Labarta, J., Cortés, U., and Suzumura, T. (2018). Fluid communities: a competitive, scalable and diverse community detection algorithm. In Complex Networks & Their Applications VI, C. Cherifi, H. Cherifi, M. Karsai, and M. Musolesi, eds. (Cham: Springer International Publishing), pp. 229–240.

Pennarun, B., Meijer, A., de Vries, E.G.E., Kleibeuker, J.H., Kruyt, F., and de Jong, S. (2010). Playing the DISC: turning on TRAIL death receptor-mediated apoptosis in cancer. Biochim. Biophys. Acta *1805*, 123–140.

Perry, N.A., Kaoud, T.S., Ortega, O.O., Kaya, A.I., Marcus, D.J., Pleinis, J.M., Berndt, S., Chen, Q., Zhan, X., and Dalby, K.N. (2019). Arrestin-3 scaffolding of the JNK3 cascade suggests a mechanism for signal amplification. Proc. Natl. Acad. Sci. U S A *116*, 810–815.

Raghavan, U.N., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. Phys. Rev. E Stat. Nonlinear Soft Matter. Phys. *76* (Pt 2), 036106.

Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D.A., and Nolan, G.P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. Science *308*, 523–529.

Schaff, J.C., Vasilescu, D., Moraru, I.I., Loew, L.M., and Blinov, M.L. (2016). Rule-based modeling with virtual cell. Bioinformatics *32*, 2880–2882.

Sekar, J.A.P., Tapia, J.J., and Faeder, J.R. (2017). Automated visualization of rule-based models. PLoS Comput. Biol. *13*, 1–23.

Smith, A.M., Xu, W., Sun, Y., Faeder, J.R., and Marai, G.E. (2012). RuleBender: integrated modeling, simulation and visualization for rule-based intracellular biochemistry. BMC Bioinformatics *13*, S3.

Spencer, S.L., Gaudet, S., Albeck, J.G., Burke, J.M., and Sorger, P.K. (2009). Non-genetic origins of cell-to-cell variability in TRAIL-induced apoptosis. Nature *459*, 428–432.

Tiger, C.F., Krause, F., Cedersund, G., Palmér, R., Klipp, E., Hohmann, S., Kitano, H., and Krantz, M. (2012). A framework for mapping, visualisation and automatic model creation of signal-transduction networks. Mol. Syst. Biol. *8*, 1–20.

Vasilescu, D., Greene, J., Schaff, J.C., Moraru, I.I., and Blinov, M.L. (2018). Molecular process diagram: a precise, scalable and compact visualization of rule-based models. bioRxiv. https://doi.org/10.1101/503359.1.

Westphal, D., Kluck, R.M., and Dewson, G. (2014). Building blocks of the apoptotic pore: how Bax and Bak are activated and oligomerize during apoptosis. Cell Death Differ. *21*, 196–205.

Xia, T., Van Hemert, J., and Dickerson, J.A. (2011). CytoModeler: a tool for bridging large-scale network analysis and dynamic quantitative modeling. Bioinformatics *27*, 1578–1580.

Yang-Yen, H.-F. (2006). Mcl-1: a highly regulated cell death and survival controller. J. Biomed. Sci. *13*, 201–204.

YIN, X.-M. (2000). Signal transduction mediated by Bid, a pro-death Bcl-2 family proteins, connects the death receptor and mitochondria apoptosis pathways. Cell Res. *10*, 161–167.

Zhou, P., Qian, L., Kozopas, K.M., and Craig, R.W. (1997). Mcl-1, a Bcl-2 family member, delays the death of hematopoietic cells under a variety of apoptosis-inducing conditions. Blood *89*, 630–643.

Zou, H., Yang, R., Hao, J., Wang, J., Sun, C., Fesik, S.W., Wu, J.C., Tomaselli, K.J., and Armstrong, R.C. (2003). Regulation of the Apaf-1/caspase-9 apoptosome by caspase-3 and XIAP. J. Biol. Chem. *278*, 8091–8098.

ISCI, Volume *23*

Supplemental Information

Interactive Multiresolution Visualization of Cellular Network Processes

Oscar O. Ortega and Carlos F. Lopez

Supplemental data items

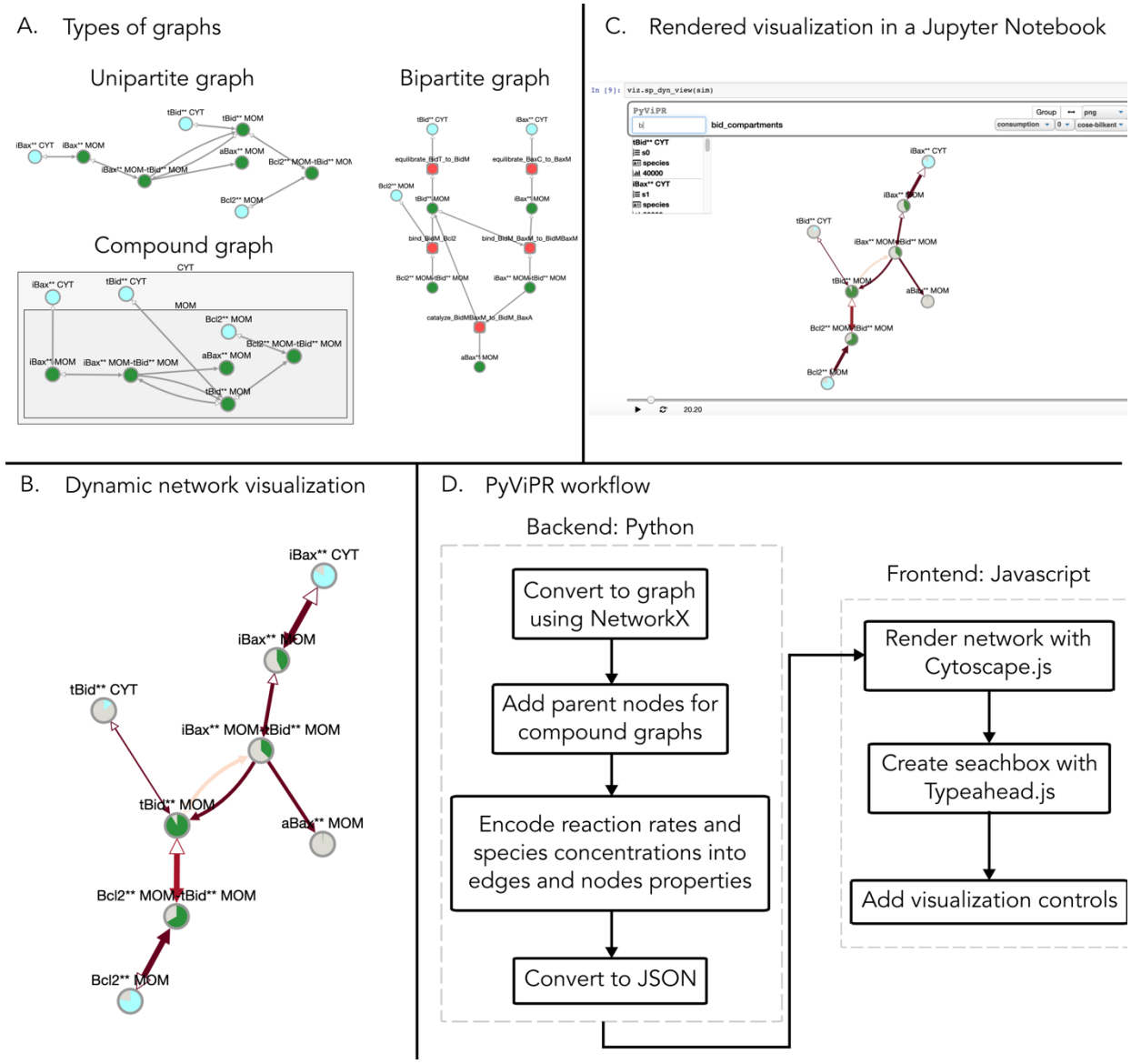


Figure S1. Network visualization modes in PyViPR, Related to Figure 1. PyViPR supports four major modes of network visualization. (A) A bipartite graph where one set of nodes represents the model species, the second set of nodes represents model model rules/reactions, and the edges connect reactant and product species with their corresponding rule/reactions. (B) A unipartite graph where each node represents a chemical species and edges represent biochemical interactions. (C) A compound graph where the nodes are grouped by the compartments/communities on which they are located. (D) Snapshot of a dynamic visualization in a unipartite graph. Nodes represent chemical species, edges represent biochemical reactions, and the pie charts inside nodes represent species concentration over time.

Figure S2. Names of functions to create model visualizations and the model components included in a network, Related to Figure 1. There are three types of model components that can be used for visualization purposes. The first one is primary nodes which correspond to model species, rules and reactions. Second is the compound nodes that include model compartments, modules/files, and communities detected by clustering algorithms. Finally, we have the dynamics information which correspond to the simulation results of a model. Marked checkboxes for each function correspond to the information displayed in a network.

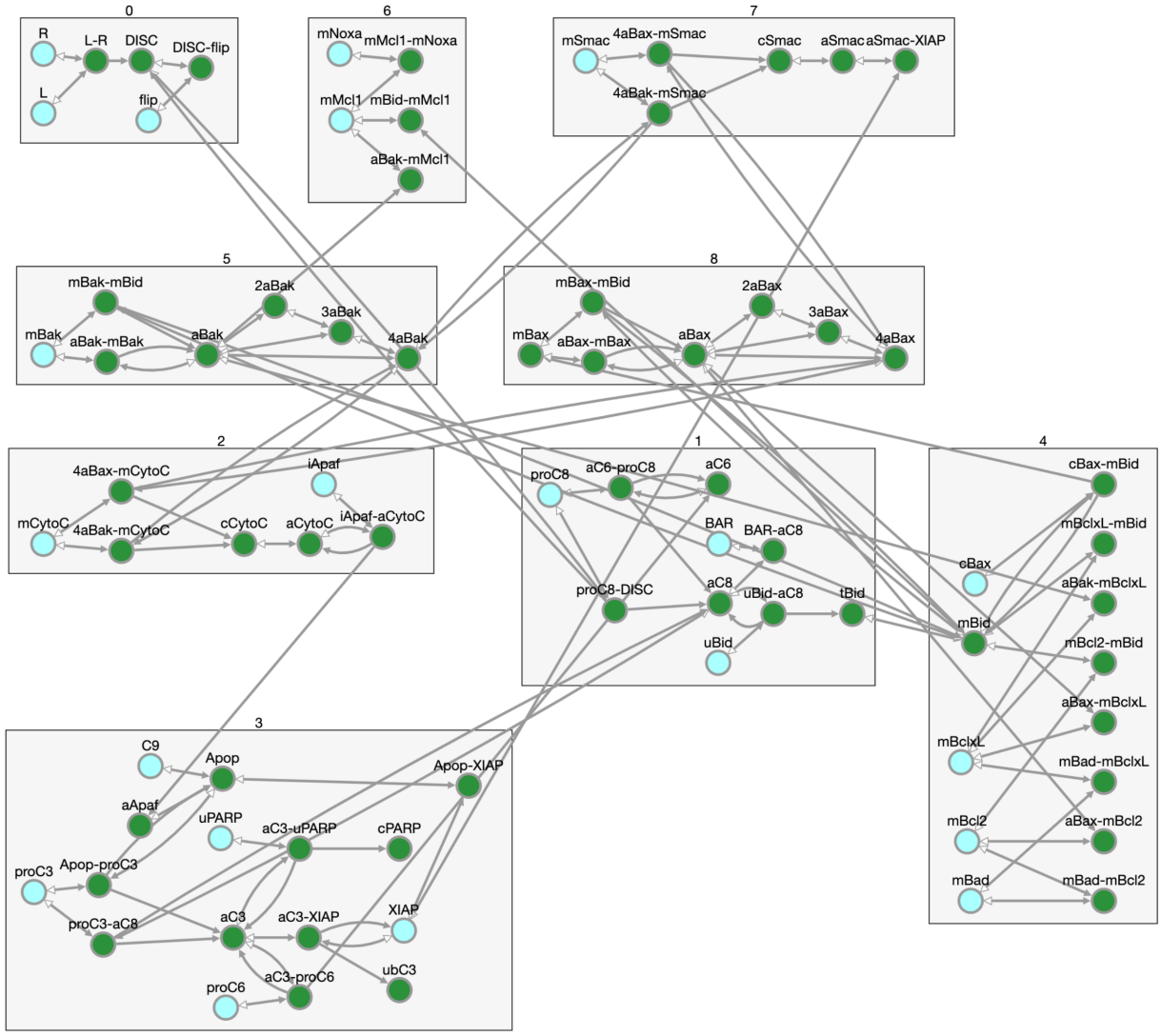


Figure S3. Communities detected in EARM, Related to Figure 2 and Table 1. Species network of EARM. Each of the nodes represents a molecular species defined in the model, and the edges depict the interactions between species. Species nodes are clustered in 9 groups. These are the communities, labeled from 0 to 8, detected by the Louvain algorithm

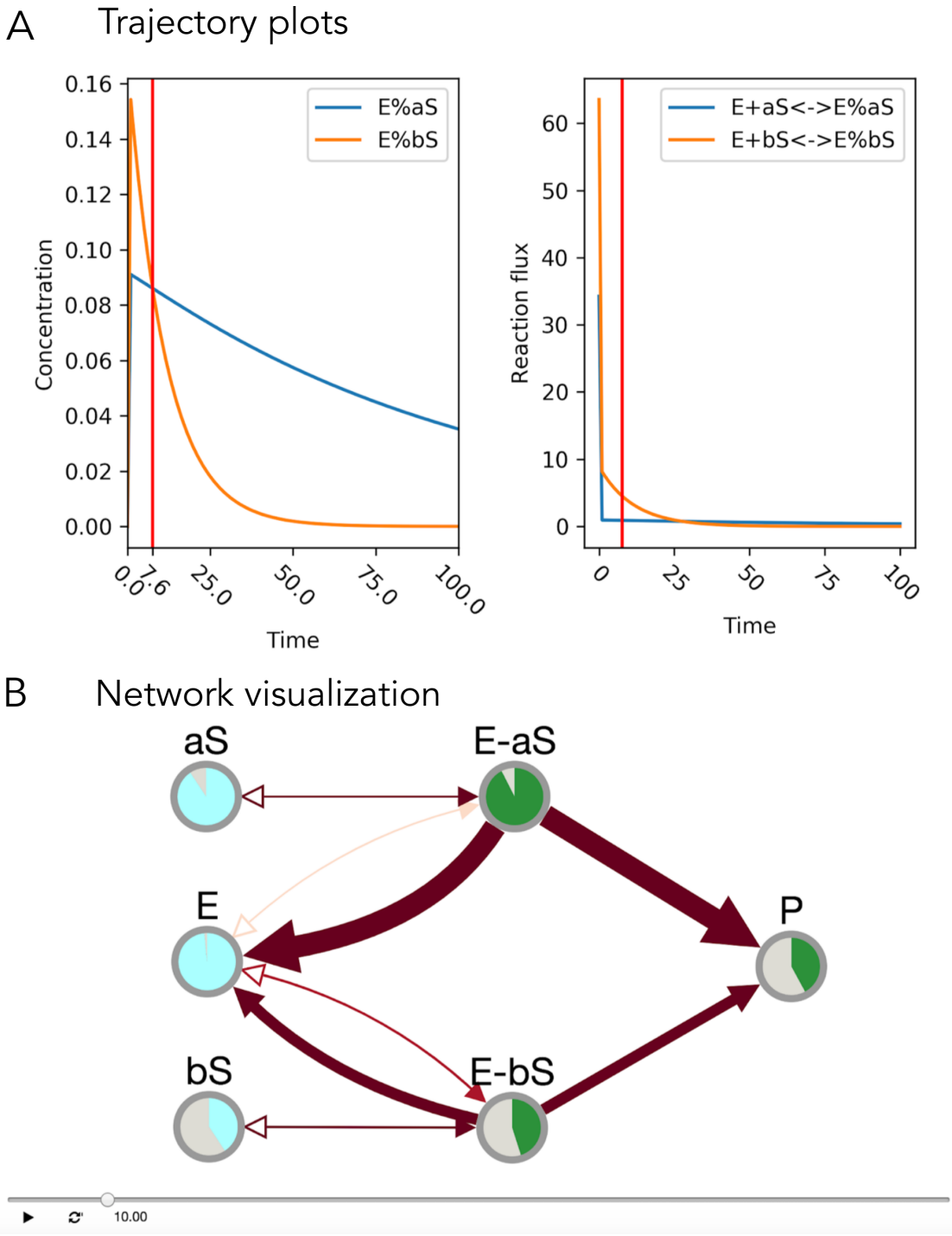


Figure S4. higher species concentration does not necessarily translate to a higher reaction flow, Related to Figure 1. A) Concentration of the complexes E-aS and E-bS over time. Substrate bS is rapidly bound to the enzyme and has a higher E-bS concentration than the E-aS complex for the first 7.6 seconds. After 7.6 seconds the concentration of substrate aS bound to the enzyme is higher than the one of substrate bS. B) Dynamic visualization of the enzymatic reaction at time point 10s. The flux from E to E-bS is higher (indicated by the darker red shade in) than the flux from E to-aS at 10 seconds.

Transparent Methods

Key Resources Table

Reagent or Resource	Source	Identifier
Data		
HeLa cells FRET reporter proteins data	(Spencer et al., 2009)	https://github.com/sorgerlab/earm
Software and Algorithms		
PyViPR	This paper	https://github.com/LoLab-VU/pyvipr
PySB	(Lopez et al., 2013)	http://pysb.org/
NetworkX	(Hagberg et al., 2008)	https://networkx.github.io
Louvain algorithm	(Blondel et al., 2008)	https://github.com/taynaud/python-louvain
Cytoscape.js	(Franz et al., 2015)	http://js.cytoscape.org

Contact for Reagent and Resource Sharing

Further information and requests for resources and reagents should be directed to and will be fulfilled by the Lead Contact, Carlos F. Lopez (c.lopez@vanderbilt.edu)

Method Details

Overview. PyViPR embeds static and dynamic network visualizations of different biochemical model components into a Jupyter Notebook (Kluyver et al., 2016). To generate these visualizations, PyViPR requires as input data a model or simulation result encoded in one of the accepted formats (See input data section below). Once a model or simulation result has been passed to one of the PyViPR functions, in the back-end PyViPR uses the Python package Networkx (Hagberg et al., 2008) to generate node-edge graphs that store information from model components (e.g. molecular species, reactions, rules) and simulation results. Species nodes can be clustered based on community detection algorithms or model compartments. Then, the NetworkX graph is converted into a JSON file that is passed to the JavaScript front-end which employs Cytoscape.js (Franz et al., 2015) and some of its extensions to render the graphs, apply layout algorithms, expand and collapse compound nodes (Dogrusoz et al., 2018), and enable the dynamic visualization of model simulation results for the visualization of networks within Jupyter Notebooks.

Input data. PyViPR currently includes three interfaces that enable the support of multiple model and graph formats: (i) PySB interface, which uses the PySB package (Lopez et al., 2013) to handle models encoded in the BioNetGen, SBML, PySCeS, E-Cell, and PySB format, (ii) Tellurium interface, which uses the Tellurium package (Medley et al., 2018) to handle models encoded in SBML and Antimony, and (iii) Graph interface, which uses NetworkX and Cytoscape.js to handle graphs encoded in GraphML, SIF, SBGN XML, Cytoscape JSON, GEXF, GML and YAML.

Output data. All visualizations are rendered as networks within a Jupyter Notebook. These networks can be locally downloaded in the following formats: PNG, SIF, GraphML and JSON.

PyViPR main visualization functions. PyViPR enables the interactive visualization of different model components as well as simulated trajectories of molecular species and reaction rates. The main visualization functions include:

- **sp_rxns_bidirectional_view(model):** Shows a bipartite network where one set of nodes are species and the other set are bidirectional reactions. Edges connect reaction nodes with their respective reactants and products species.
- **sp_view(model):** Shows the unipartite network of interacting species.
- **projected_bireactions_view(model):** Shows the unipartite network of reactions projected from the bipartite species-reactions graph.
- **sp_comm_louvain_view(model):** Shows the unipartite network of interacting species grouped by the Louvain community detection algorithm.
- **sp_dyn_view(model):** Shows a species network. Edges size and color are updated according to reaction rate values, and nodes pie charts slices are updated according to the concentration of species.
- **cluster_rxns_by_rules_view(model):** Shows the unipartite graph of the interactions between the reactions in a model. Reaction nodes are grouped by the rules that generated them.
- **highlight_nodes_view(species, reactions):** Highlights the species and/or reactions passed as arguments.

A more detailed description of these and other visualization functions can be found on the PyViPR documentation website (https://pyvipr.readthedocs.io/)

Bipartite projection to a species graph

The following algorithm describes how PyViPR projects a species-reactions bipartite graph to a species-only graph. Briefly, PyViPR links two species nodes if they are connected through a reaction node, with the exception of species nodes that are both reactants of the same reaction as edges should only connect reactant species nodes with products species nodes. The projection of catalytic reactions results in the enzyme node having two edges to the enzyme-substrate complex node: one to represent the reversible substrate binding reaction and a second to represent the catalytic reaction

Algorithm 1: Computing species projected network

```
input  : Bipartite graph, species nodes to project onto, model reactions
output : A species projected graph
sp_graph = graph(species_nodes);
foreach node in species_nodes do
    neighbors2 ← [ ];
    foreach node1 in node.neighbors do
        reactants ← reactions[node1].reactants;
        foreach node2 in node1.neighbors do
            if node != node 2 and node2 not in reactants then
                neighbors2.add(node2)
            end
        end
    end
    foreach n in neighbors2 do
        sp_graph.add_edge(node, node2)
    end
end
```

Model calibration

For the calibration of EARM, nominal values for rate constants were set to their published values in EARM 2.0 (Lopez et al., 2013). All rate constants were allowed to change two orders of magnitude above and below their nominal values, indicating a lack of knowledge about the likely parameter values. Experimental time-courses of the initiator caspase reporter protein (IC-RP), mitochondrial inter-membrane space reporter protein (IMS-RP), and effector caspase reporter protein (EC-RP) were used from previously published data (Spencer et al., 2009). In the model, tBid, cytosolic Smac, and cleaved PARP were fit to the data for IC-RP, IMS-RP, and EC-RP, respectively. Model calibration was then performed using the simplePSO package (Pino, 2019), which is a python implementation of the Particle Swarm Optimization algorithm (Kennedy and Eberhart, 1995). The fit of simulated trajectories to experimental data was measured using the sum of the squared differences:

$$\chi = \sum_{t}\sum_{i} \frac{1}{2\sigma_{data}^{2}} \left[x_{model}^{i}(t;\Theta) - x_{data}^{i}(t)\right]^{2}$$

Where t is the time span of the simulation and experiments, $\Theta = (\theta_1, \ldots, \theta_n)$ are the parameters of the model, x_{model}^{i} are the simulations of the model under condition i and x_{data}^{i} is the experimental data under condition i.

We first ran PSO 100 times to determine a reasonable cost function threshold to consider that a calibrated parameter is a good fit. We chose the parameter set with the lowest function, which corresponds to a value of 2.8, and visually inspected that the fit was good. Then, we ran PSO 10000 times, and only kept parameter sets that had a cost function of 2.8 or less.

Parameter selection for analysis

To obtain the two maximally different parameter sets from the 6572 calibrated parameter sets, we first standardize the kinetic parameter values as kinetic parameters with a variance that is order of magnitudes larger than others might dominate the distance function and lead to parameters that are mostly different at that specific kinetic parameter. To standardize the values of a kinetic parameter we remove the mean and scale to unit variance:

$$\boldsymbol{z} = (\boldsymbol{x} - \boldsymbol{u})/\boldsymbol{s}$$

Where x is a value of a specific kinetic parameter, u and s are the mean and the standard deviation of the specific kinetic parameter, respectively.

After standardizing the kinetic parameter values, we use the Euclidean metric (eq. 1) to calculate the pairwise distances between the 6572 calibrated parameter sets, and then choose the two parameter sets that yield the largest distance.

$$(1) \quad d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{1}$$

Where $\mathrm{p} = (\mathrm{p}_1, \mathrm{p}_2, \ldots, \mathrm{p}_\mathrm{n})$ and $\mathrm{q} = (\mathrm{q}_1, \mathrm{q}_2, \ldots, \mathrm{q}_\mathrm{n})$ are two parameter sets, and n is the number of kinetic parameters in the model.

Supplemental References

Spencer SL, Gaudet S, Albeck JG, Burke JM, Sorger PK. Non-genetic origins of cell-to-cell variability in TRAIL-induced apoptosis. Nature.2009;459(7245):428–432. doi:10.1038/nature08012

Lopez CF, Muhlich JL, Bachman JA, Sorger PK. Programming biological models in Python using PySB. Molecular Systems Biology. 2013;9(646):1–19.doi:10.1038/msb.2013.1

Hagberg AA, Schult DA, Swart PJ. Exploring network structure, dynamics, and function using NetworkX. Proceedings of the 7th Python in Science Conference(SciPy). 2008;(SciPy):11–15. doi:10.1016/j.jelectrocard.2010.09.003.

Blondel VD, Guillaume JL, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment. 2008;2008(10). doi:10.1088/1742-5468/2008/10/P10008.

Franz M, Lopes CT, Huck G, Dong Y, Sumer O, Bader GD. Cytoscape.js: A graph theory library for visualisation and analysis. Bioinformatics.2015;32(2):309–311. doi:10.1093/bioinformatics/btv557.

Kluyver T, Ragan-Kelley B, Pérez F, Granger BE, Bussonnier M, Frederic J, et al. Jupyter Notebooks - a publishing format for reproducible computational workflows. In: ELPUB; 2016.

Dogrusoz U, Karacelik A, Safarli I, Balci H, Dervishi L, Siper MC. Efficient methods and readily customizable libraries for managing complexity of large networks. PLOS ONE. 2018;13(5):1–18. doi:10.1371/journal.pone.0197238.

Medley JK, Choi K, König M, Smith L, Gu S, Hellerstein J, et al. Tellurium notebooks-An environment for reproducible dynamical modeling in systems biology. PLOS Computational Biology. 2018;14(6):1–24.doi:10.1371/journal.pcbi.1006220.

Pino J. LoLab-VU/ParticleSwarmOptimization: simplePSO release 1.0; 2019.Available from: https://doi.org/10.5281/zenodo.2612913.

Kennedy J, Eberhart R. Particle swarm optimization. Neural Networks, 1995 Proceedings, IEEE International Conference on. 1995;4:1942–1948 vol.4.doi:10.1109/ICNN.1995.488968.