GRNsight: a web application and service for visualizing models of small- to medium-scale gene regulatory networks

Kam D Dahlquist, John David N Dionisio, Ben G Fitzpatrick, Nicole A Anguiano, Anindita Varshneya, Britain J Southwick, Mihir Samdarshi

GRNsight is a web application and service for visualizing models of gene regulatory networks (GRNs). A gene regulatory network consists of genes, transcription factors, and the regulatory connections between them which govern the level of expression of mRNA and protein from genes. The original motivation to create GRNsight came from our efforts to perform parameter estimation and forward simulation of the dynamics of a differential equations model of a small GRN with 21 nodes and 31 edges. We wanted a quick and easy way to visualize the weight parameters from the model which represent the direction and magnitude of the influence of a transcription factor on its target gene, so we created GRNsight. GRNsight automatically lays out either an unweighted or weighted network graph based on an Excel input spreadsheet containing an adjacency matrix where regulators are named in the columns and target genes in the rows. When a user uploads a spreadsheet with an unweighted adjacency matrix, GRNsight automatically lays out the graph using black lines and pointed arrowheads. When a user uploads a spreadsheet with a weighted adjacency matrix, GRNsight uses pointed and blunt arrowheads, and colors the edges and adjusts their thicknesses based on the sign (positive for activation or negative for repression) and magnitude of the weight parameter. GRNsight is written in JavaScript, with diagrams facilitated by D3.js, a data visualization library. Node.js and the Express framework handle server-side functions. GRNsight's diagrams are based on D3.js's force graph layout algorithm, which was then extensively customized to support the specific needs of GRN visualization. Nodes are rectangular and support gene labels of up to 12 characters. The edges are arcs, which become straight lines when the nodes are close together. Self-regulatory edges are indicated by a loop on the lower-right side of a node. When a user mouses over an edge, the numerical value of the weight parameter is displayed. Visualizations can be modified by sliders that adjust D3.js's force graph layout parameters and through manual node dragging. GRNsight is best-suited for visualizing networks of fewer than 35 nodes and 70 edges, although it accepts networks of up to 75 nodes or 150 edges. Although originally designed for GRNs, GRNsight has general applicability for displaying any small, unweighted or weighted network with directed edges for systems biology or other application domains. GRNsight serves as an example of

following and teaching best practices for scientific computing, using an open and test-driven development model with rigorous documentation of requirements and issues on GitHub. An exhaustive unit testing framework using Mocha and the Chai assertion library consists of over 130 automated unit tests that examine over 520 test files to ensure that the program is running as expected. GRNsight is available under the open source BSD license at http://dondi.github.io/GRNsight/ .

GRNsight: a web application and service for visualizing models of small- to medium-scale gene regulatory networks

Kam D. Dahlquist[1*], John David N. Dionisio[2], Ben G. Fitzpatrick[3], Nicole A. Anguiano[2], Anindita Varshneya[1], Britain J. Southwick[2], Mihir Samdarshi[1]

[1]Department of Biology, [2]Department of Electrical Engineering and Computer Science, [3]Department of Mathematics, Loyola Marymount University, 1 LMU Drive, Los Angeles, CA 90045 USA

[*]Corresponding author:
Kam D. Dahlquist
Department of Biology
Loyola Marymount University
1 LMU Drive, MS8888
Los Angeles, CA 90045 USA
E-mail: kdahlquist@lmu.edu
Tel: 1-310-338-7697

Link to web application: http://dondi.github.io/GRNsight/

Link to code repository: https://github.com/dondi/GRNsight

Abstract

GRNsight is a web application and service for visualizing models of gene regulatory networks (GRNs). A gene regulatory network consists of genes, transcription factors, and the regulatory connections between them which govern the level of expression of mRNA and protein from genes. The original motivation to create GRNsight came from our efforts to perform parameter estimation and forward simulation of the dynamics of a differential equations model of a small GRN with 21 nodes and 31 edges. We wanted a quick and easy way to visualize the weight parameters from the model which represent the direction and magnitude of the influence of a transcription factor on its target gene, so we created GRNsight. GRNsight automatically lays out either an unweighted or weighted network graph based on an Excel input spreadsheet containing an adjacency matrix where regulators are named in the columns and target genes in the rows. When a user uploads a spreadsheet with an unweighted adjacency matrix, GRNsight automatically lays out the graph using black lines and pointed arrowheads. When a user uploads a spreadsheet with a weighted adjacency matrix, GRNsight uses pointed and blunt arrowheads, and colors the edges and adjusts their thicknesses based on the sign (positive for activation or negative for repression) and magnitude of the weight parameter. GRNsight is written in JavaScript, with diagrams facilitated by D3.js, a data visualization library. Node.js and the Express framework handle server-side functions. GRNsight's diagrams are based on D3.js's force graph layout algorithm, which was then extensively customized to support the specific needs of GRN visualization. Nodes are rectangular and support gene labels of up to 12 characters. The edges are arcs, which become straight lines when the nodes are close together. Self-regulatory edges are indicated by a loop on the lower-right side of a node. When a user mouses over an edge, the numerical value of the weight parameter is displayed. Visualizations

can be modified by sliders that adjust D3.js's force graph layout parameters and through manual node dragging. GRNsight is best-suited for visualizing networks of fewer than 35 nodes and 70 edges, although it accepts networks of up to 75 nodes or 150 edges. Although originally designed for GRNs, GRNsight has general applicability for displaying any small, unweighted or weighted network with directed edges for systems biology or other application domains. GRNsight serves as an example of following and teaching best practices for scientific computing, using an open and test-driven development model with rigorous documentation of requirements and issues on GitHub. An exhaustive unit testing framework using Mocha and the Chai assertion library consists of over 130 automated unit tests that examine over 520 test files to ensure that the program is running as expected. GRNsight is available under the open source BSD license at http://dondi.github.io/GRNsight/.

Introduction

GRNsight is a web application and service for visualizing models of small- to medium-scale gene regulatory networks (GRNs). A gene regulatory network consists of genes, transcription factors, and the regulatory connections between them which govern the level of expression of mRNA and protein from genes. Our group has developed a MATLAB program to perform parameter estimation and forward simulation of the dynamics of an ordinary differential equations model of a medium-scale GRN with 21 nodes and 31 edges (Dahlquist et al., 2015; *http://kdahlquist.github.io/GRNmap/*). GRNmap accepts a Microsoft Excel workbook as input, with multiple worksheets specifying the different types of data needed to run the model. For compactness, the GRN itself is specified by a worksheet that contains an adjacency matrix where regulators are named in the columns and target genes in the rows. Each cell in the matrix contains a "0" if there is no regulatory relationship between the regulator and target, or a "1" if there is a regulatory relationship between them. The GRNmap program then outputs the estimated weight parameters in a new worksheet containing an adjacency matrix where the "1's" are replaced with a real number that is the weight parameter, representing the direction (positive for activation or negative for repression) and magnitude of the influence of the transcription factor on its target gene (Dahlquist et al., 2015). Although MATLAB has graph layout capabilities, we wanted a way for novice and experienced biologists alike to quickly and easily view the network graph corresponding to the matrix without having to create or modify MATLAB code.

Pavlopoulos et al. (2015) have recently reviewed the types, trends, and usage of visualization tools available for genomics and systems biology, listing a total of 47 stand-alone

and web-based tools for network analysis. With such a large number of tools available, it would be reasonable to expect that one already exists that could fulfill our needs. However, despite this diversity of tools, each had properties that limited their use for us. For example, some were hard coded for a different type of network (e.g., metabolic or signaling pathways, protein-protein interaction networks) or were designed for visualization and analysis of much larger networks than the ones in which we were interested. None would readily accept an adjacency matrix with the GRNmap specifications as input without some manipulation of the data format. Many required installation of stand-alone software, and/or had a steep learning curve. As an illustration of this, Pavlopoulos et al. (2015) showed that the open source software, Cytoscape (Shannon et al., 2003; Smoot et al., 2011) had the highest citation count in the Scopus database, as it is widely recognized as the "best-in-class" tool for viewing and analyzing large networks for systems biology research. However, while Cytoscape is flexible in terms of what types of network representations it accepts as input (SIF, NNF, GML, XGMML, SBML, BioPAX, PSI-MI, GraphML, cf. *http://manual.cytoscape.org/en/latest/Supported_Network_File_Formats.html#supported-network-file-formats*), its basic "unformatted table files" format expects the network to be represented in a list of pairwise interactions between two nodes instead of as an adjacency matrix, requiring a GRNmap user to convert the file external to the program. Furthermore, Cytoscape must be installed on a user's computer. Finally, because it is powerful and has a lot of features, there is a somewhat steep learning curve before a novice user can begin to visualize networks. Multiple settings must be learned and selected to generate a display that properly fits a use case; it is not possible to just "load into Cytoscape and go." Another open source application, Gephi (Bastian, Heymann, and Jacomy, 2009), is a general graph visualization tool

that does accept an adjacency matrix in .csv format (among a wide range of supported formats, cf. *https://gephi.org/users/supported-graph-formats/csv-format/*), but again requires download and installation of the software and has a complex feature set. Because GRNmap itself is complex software targeted both at experienced biology investigators and novice undergraduate users in a Biomathematical Modeling course, we wanted to limit the need to install and learn additional visualization software. Reducing the cognitive load required for using the software would allow users to focus their attention on understanding the biological results of the model.

We also saw the creation of a new tool as an opportunity to serve as a model for best practices for software development in bioinformatics (Schultheiss, 2011; Wilson et al., 2014), simultaneously following and teaching these practices to the primary developers who were all undergraduates. Following the philosophy of "do one thing well" (*http://onethingwell.org/post/457050307/about-one-thing-well*), we wanted to prioritize rendering a small- to medium-scale gene regulatory networks both easily and well. It was more important for us to create a tool that is specifically tailored to the visualization of these sized GRNs, and not every possible graph from every possible application domain. This specific tailoring also included minimizing any startup, onboarding, or overhead time. Thus we had the following requirements for GRNsight. It should:

- Exist as a web application without the need to download and install specialized software;
- Accept an input file in Microsoft Excel format (.xlsx);
- Read a weighted or unweighted adjacency matrix where the regulatory transcription factors are in columns and the target genes are in rows;

- Automatically lay out and display unweighted and weighted network graphs in a way that is familiar to biologists.

GRNsight fulfills these requirements as described below.

Materials and Methods

Input Data

GRNsight automatically lays out the network graph specified by an adjacency matrix contained within a worksheet named "network" or "network_optimized_weights" in a Microsoft Excel workbook (.xlsx). It was designed to accept workbooks seamlessly from the MATLAB gene regulatory network modeling program, GRNmap; however, the expected input format is general and is not dependent on GRNmap. Detailed documentation for the expected input file format is found on the GRNsight Documentation page: *http://dondi.github.io/GRNsight/documentation.html*.

GRNsight can automatically lay out either an unweighted or weighted network graph specified by an adjacency matrix where regulators are named in the columns and target genes in the rows. Note that regulators (regulatory transcription factors) are themselves encoded by genes and will be referred to as such. The adjacency matrix can be either symmetric (with the exact same genes named in both the columns and rows) or asymmetric (additional genes in either the columns or rows or both). For an unweighted network, each cell in the matrix should contain a "0" if there is no regulatory relationship between the regulator and target, or a "1" if there is a regulatory relationship between them (Fig. 1). In a weighted network, the "1's" are replaced with a real number that is the weight parameter (Fig. 2). Positive weights indicate activation of

the target gene by the regulator, and negative weights indicate repression of the target gene by the regulator.

GRNsight is designed to visualize small- to medium-scale GRNs, not the entire gene regulatory network for an organism. The bounding box for display of the graph has a fixed size. Currently, it is recommended that the user upload networks with no more than 35 unique genes (nodes) or 70 edges. A warning is given upon upload of a network with 50-74 nodes or 71-99 edges, although the network graph will still display. If the user attempts to upload a network of 75 or more nodes or 100 or more edges, the graph does not display, and an error message will be returned.

Architecture

GRNsight has a service-oriented architecture, consisting of separate server and web client components (Fig. 3). The server provides a web API (application programming interface) that accepts a Microsoft Excel workbook (.xlsx) file via a POST request and converts it into a corresponding JSON (JavaScript Object Notation) representation. Conversion is accomplished by first parsing the .xlsx file using the node-xlsx library (*https://github.com/mgcrea/node-xlsx*) then mapping the translated worksheet cells into JSON. It also provides demonstration graphs already in this JSON format, without requiring a spreadsheet upload. The web client provides a graphical user interface for visualizing the JSON graphs provided by the server, whether the graphs are parsed from uploaded Excel workbooks or provided directly by the server's demos. As an additional layer of customization, the graphical interface provided by the web client can be embedded in any web page using the standard *iframe* element. This is the mechanism used in deploying the production and beta versions of the software on *https://dondi.github.io/GRNsight*.

Figure 3 illustrates this architecture and the interactions of the components. Documentation for how GRNsight is specifically deployed, including autonomous production and beta versions, can be found on the GRNsight wiki (*https://github.com/dondi/GRNsight/wiki/Server-Setup*).

GRNsight is an open source project and is itself built using other open source software. Server-side components are implemented with Node.js and the Express framework (Brown 2014). Graph visualization is facilitated by the Data-Driven Documents JavaScript library (D3.js; Bostock, Ogievetsky, and Heer, 2011). D3.js provides data mapping and layout routines which GRNsight heavily customizes in order to achieve the desired graph visualization. The resulting graph is a Scalable Vector Graphics (SVG) drawing in which D3.js maps gene objects from the JSON representation provided by the web API server onto labeled rectangles. Edge weights are mapped into Bezier curves. The resulting graph is interactive, initially using D3.js's *force graph layout* algorithm to automatically determine the positions of the gene rectangles. The user can then drag the rectangles to improve the graph's layout. Customizations to the graph display are described further below. While Cytoscape.js (Franz et al., 2016) is also an open source network visualization engine, we chose to build GRNsight with D3.js because of the future possibility of implementing other D3.js visualizations and because of the prior familiarity with the D3.js library by one of the co-authors.

Graph Customizations

GRNsight's diagrams are based on D3.js's force graph layout algorithm (Bostock, Ogievetsky, and Heer, 2011), which was then extensively customized to support the specific needs of biologists for GRN visualization. D3.js's baseline force graph implementation had round, unlabeled nodes and undirected, straight-line edges. The following customizations were

made for the nodes: (a) the nodes were made rectangular; (b) a label of up to 12 characters was added; (c) node size was varied, depending on the size of the label.

Customizations were also made for the edges. Instead of undirected, straight line segments, the edges display as directed edges. They are implemented as Bezier curves that straighten when nodes are close together and curve when nodes are far apart. A special case was added to form a looping edge if a node regulated itself. When an unweighted adjacency matrix is uploaded, all edges are displayed as black with pointed arrowheads. When a weighted adjacency matrix is uploaded, edges are further customized based on the sign and magnitude of the weight parameter. As is common practice in biological pathway diagrams (Gostner et al., 2014), activation (for positive weights) is represented by pointed arrowheads, and repression (for negative weights) is represented by a blunt end marker, i.e., a line segment perpendicular to the edge. The thickness of the edge also varies based on the magnitude of the absolute value of the weight. Larger magnitudes have thicker edges and smaller magnitudes have thinner edges. The way that GRNsight determines the edge thickness is as follows: GRNsight divides all weight values by the absolute value of the maximum weight in the adjacency matrix to normalize all the values to between zero and 1. GRNsight then adjusts the thickness of the lines to vary continuously from the minimum thickness (for normalized weights near zero) to maximum thickness (normalized weight of 1). The color of the edge also imparts information about the regulatory relationship. Edges with positive normalized weight values from 0.05 to 1 are colored magenta; edges with negative normalized weight values from -0.05 to -1 are colored cyan. Edges with normalized weight values between -0.05 and 0.05 are colored grey to emphasize that their normalized magnitude is near zero and that they have a weak influence on the target gene. When a user mouses over an edge, the numerical value of the weight parameter is displayed. When the

user drags nodes to customize his or her view of the network, edges adapt their anchor points to the movements of the nodes.

User Interface

The GRNsight user interface includes a menu/status bar and sliders that adjust D3.js's force graph layout parameters. Figure 4 provides an annotated screenshot of the user interface, highlighting its primary features. Users can move force graph parameter sliders to refine the automated visualization. Nodes have a *charge*, which repels or attracts other nodes. The *charge distance* determines at what range a node's charge will affect other nodes. The *link distance* determines the minimum distance maintained between nodes. *Gravity* determines the strength of the force holding the nodes to the center of the graph. Sliders can be locked to prevent changes and also reset to default values. Graph visualizations can also be modified through manual node dragging. Design decisions for the user interface were driven by applicable interaction design guidelines and principles (Nielsen 1993; Shneiderman et al., 2016; Norman 2013) in alignment with the mental model and expectations of the target user base, consisting primarily of biologists, both novice and experienced.

Test-driven Development

GRNsight follows an open development model with rigorous documentation of requirements and issues on GitHub. We have implemented an exhaustive unit testing framework using Mocha (*https://mochajs.org*) and the Chai assertion library (*http://chaijs.com*) to perform test-driven development where unit tests are written before new functionality is coded (Martin 2008). This framework consists of over 130 automated unit tests that examine over 520 test files

to ensure that the program is running as expected. Table 1 shows the test suite's coverage report, as generated by Istanbul (*https://gotwarlost.github.io/istanbul/*).

Error and warning messages have a three-part framework that informs the user what happened, the source of the problem, and possible solutions. This structure follows the alert elements recommended by user interface guideline documents such as the OS X Human Interface Guidelines (*https://developer.apple.com/library/mac/documentation/UserExperience/Conceptual/OSXHIGuidelines/WindowAlerts.html*). For example, GRNsight returns an error when the spreadsheet is formatted incorrectly or the maximum number of nodes or edges is exceeded.

Availability

GRNsight is available at *http://dondi.github.io/GRNsight/*and is compatible with Google Chrome version 43.0.2357.65 or higher and Mozilla Firefox version 38.0.1 or higher on the Windows 7 and Mac OS X operating systems. Web site content is available under the Creative Commons Attribution Non-Commercial Share Alike 3.0 Unported License. GRNsight code is available under the open source BSD license from our GitHub repository *https://github.com/dondi/GRNsight*. Every user's submitted data are private and not viewable by anyone other than the user. Uploaded data reside as temporary files and are deleted from the GRNsight server during standard operating system file cleanup procedures. A Google Analytics page view counter was implemented on 18 September 2014, and a file upload counter was added on 13 April 2015. From these start dates and as of 14 May 2016, the GRNsight home page has been accessed 2019 times, and 1530 files have been uploaded and viewed with GRNsight. Of these 1530 files, an estimated 65 were uploaded by users outside of our group.

Results and Discussion

We have successfully implemented GRNsight, a web application and service for visualizing small- to medium-scale gene regulatory networks, fulfilling our four requirements:

- It exists as a web application without the need to download and install specialized software;
- It accepts an input file in Microsoft Excel format (.xlsx);
- It reads a weighted or unweighted adjacency matrix where the regulatory transcription factors are in columns and the target genes are in rows;
- It automatically lays out and displays unweighted and weighted network graphs in a way that is familiar to biologists.

GRNsight Facilitates Interpretation of GRN Model Results

GRNsight facilitates the biological interpretation of unweighted and weighted gene regulatory network graphs. Our discussion focuses on two of the demonstration files provided in the user interface, Demo #3: Unweighted GRN (21 genes, 31 edges) and Demo #4: Weighted GRN (21 genes, 31 edges, Schade et al. 2004 data). These two files describe gene regulatory networks from budding yeast, *Saccharomyces cerevisiae*, correspond to supplementary data published by Dahlquist et al. (2015), and when displayed by GRNsight, represent interactive versions of Figures 1 and 8 of that paper, respectively.

Figure 5 gives a side-by-side view of the same adjacency matrices laid out by GRNsight and by hand. Figures 5A, 5B, and 5C are derived from Demo #3: Unweighted GRN (21 genes, 31 edges), and Figures 5D, 5E, and 5F are derived from Demo #4: Weighted GRN (21 genes, 31

edges, Schade et al. 2004 data). Figures 5A and 5D show examples of the automatic layout performed by GRNsight. Figures 5C and 5F show the same adjacency matrices laid out by hand in Adobe Illustrator, corresponding to Figure 1 and Figure 8 of Dahlquist et al. (2015), respectively. Figures 5B and 5E started with the automatic layout from GRNsight and then were manually manipulated from within GRNsight to lay them out similarly to Figures 5C and 5F, respectively. The use of GRNsight represents a substantial time savings compared to creating the same figures entirely by hand and allows the user to try multiple arrangements of the nodes quickly and easily. Note that this type of "by hand" manipulation of graphs is most useful for small- to medium-scale networks, the kind that GRNsight is designed to display, and would not be appropriate for large networks.

Viewing the unweighted network (Fig. 5A, B, C) allows one to make observations about the network structure (Dahlquist et al., 2015). For example, YAP6 has the highest in-degree, being regulated by six other transcription factors. RAP1 has the highest out-degree of five, regulating four other transcription factors and itself. Four genes, AFT1, NRG1, RAP1, and YAP6, regulate themselves. Many of the transcription factors are involved in regulatory chains, with the longest including five nodes originating at SKN7. There are several other 4-node chains that originate at CIN5, MAC1, PHD1, SKN7, and YAP1. Finally, there are two rather complex feedforward motifs involving CIN5, ROX1, and YAP6 and SKN7, YAP1, and ROX1 (Dahlquist et al., 2015).

The networks with colored edges (Fig. 5D, E, F) display the results of a mathematical model, where the expression levels of the individual transcription factors were modeled using mass balance ordinary differential equations with a sigmoidal production function and linear degradation (Dahlquist et al., 2015). Each equation in the model included a production rate, a

degradation rate, weights that denote the magnitude and type of influence of the connected transcription factors (activation or repression), and a threshold of expression. The differential equation model was fit to published yeast cold shock microarray data from Schade et al. (2004) using a penalized nonlinear least squares approach. The visualization produced by GRNsight is displaying the results of the optimized weight parameters. Positive weights > 0 represent an activation relationship and are shown by pointed arrowheads. One example is that CIN5 activates the expression of MSN1. Negative weights < 0 represent a repression relationship and are shown by a blunt arrowhead. One example is that ABF1 represses the expression of MSN1. The thicknesses of the edges also vary based on the magnitude of the absolute value of the weight, with larger magnitudes having thicker edges and smaller magnitudes having thinner edges. In Figures 5D, E, and F, the thickest edge corresponds to the repression of the expression of MSN1 by ABF1 because the absolute value of its weight parameter (-2.97) has the highest magnitude out of all the weights (Dahlquist et al., 2015).

The color of the edge also imparts information about the regulatory relationship. Edges with positive normalized weight values from 0.05 to 1 are colored magenta (10 edges in this example); edges with negative normalized weight values from -0.05 to -1 are colored cyan (16 edges in this example). Edges with normalized weight values between -0.05 and 0.05 are colored grey to indicate that their normalized magnitude is near zero and that they have a weak influence on the target gene (5 edges in this example). The grey color de-emphasizes the weak relationships to the eye, thus emphasizing the stronger colored relationships.

Because of this visualization of the weight parameters, one can make some interesting observations about the behavior of the network (Dahlquist et al., 2015). The expression of several genes is controlled by a balance of activation and repression by different regulators. For

example, the expression of MSN1 is strongly activated by CIN5, but even more strongly repressed by ABF1. Furthermore, some transcription factors act both as activators of some targets and repressors of other targets. For example, RAP1 activates the expression of MSN4 and RPH1, but represses the expression of AFT1, HSF1, and itself. RAP1 is known to act as both an activator and a repressor (Shore and Nasmyth, 1987). Thus, GRNsight enables one to interpret the weight parameters more easily than one could from the adjacency matrix alone.

Note that the nodes in Figure 5F are also colored in the style of GenMAPP 2 (Salomonis et al. 2007), based on the time course of expression of that gene in the Schade et al. (2004) microarray data (stripes from left to right, 10, 30, and 120 minutes of cold shock, with magenta representing a significant increase in expression relative to the control at time 0, cyan representing a significant decrease in expression relative to the control, and grey representing no significant change in expression relative to the control). This feature has not yet been implemented in GRNsight, but is currently under development for Version 2.

These observations made by direct inspection of the GRNsight graph are for a relatively small GRN of 21 genes and 31 edges and become more difficult as nodes and edges are added. For much larger networks, a more powerful graph analysis tool such as Cytoscape (Shannon et al., 2003; Smoot et al., 2011) or Gephi (Bastian, Heymann, and Jacomy, 2009) is warranted. However, for small networks in the range of 15-35 nodes, GRNsight fulfills a need to quickly and easily view and manipulate them. The GRN modeled in Dahlquist et al. (2015) and displayed in Figure 5 was derived by hand from the Lee et al. (2002) and Harbison et al. (2004) datasets generated by chromatin immunoprecipitation followed by microarray analysis. We have also used GRNsight to display GRNs derived from the YEASTRACT database (Teixeira et al., 2014), whose own display tool is static, displaying regulators and targets in two rows.

Instructions for viewing YEASTRACT-derived GRNs can be found on the GRNsight documentation page.

While GRNsight was designed originally for viewing gene regulatory networks, it is not specific for that kind of data. As long as the text strings used as identifiers for the "regulators" and "targets" match, it can be used to visualize any small, unweighted or weighted network with directed edges for systems biology or other application domains.

GRNsight Development Follows Best Practices for Scientific Computing

Veretnik, Fink, and Bourne (2008) lament and Schultheiss et al. (2011) document that some computational biology resources, especially web servers, lack persistence and usability, leading to an inability to reproduce results. With that in mind, we have consciously followed best practices for scientific computing documented by Wilson et al. (2014) and for providing a web resource (Schultheiss, 2011). We have followed an open development model since the project inception in January 2014, with our code available under the open source BSD license at the public GitHub repository, where we also track requirements, issues, and bugs. Indeed, our project stands on the shoulders of other open source tools. Our unit-testing framework provides confidence that the code works as expected. Detailed documentation for users (web page) and developers (wiki) are provided. Demo data are also provided so users have both an example of how to format input files and can see how the software should perform. We are committed to continue development of the GRNsight resource, fixing bugs and improving the software by adding features. The lead authors (Dahlquist, Dionisio, and Fitzpatrick) are all tenured faculty, overseeing the design, code, testing, and documentation of GRNsight and providing continuity to the project. Together we have mentored the undergraduates (Anguiano, Varshneya, Southwick, and Samdarshi) who had primary responsibility for coding, testing, and documentation, while

also being full partners in the design of the software. A pipeline has been established for onboarding new members to the project, also providing continuity. Lawlor and Walsh (2015) detail some of the same issues of reliability and reproducibility in bioinformatics software referred to by Wilson et al. (2014). Lawlor and Walsh (2015) conclude that the ideal way to bring software engineering values into bioinformatics research projects is to establish separate specialists in bioinformatics engineering. We disagree. Through GRNsight, we have shown how best practices can be taught to undergraduates concomitant with training in bioinformatics, as we have shown previously with Master's level students (Dionisio and Dahlquist, 2008).

Conclusions

We have successfully implemented GRNsight, a web application and service for visualizing small- to medium-scale gene regulatory networks. GRNsight accepts an input file in Microsoft Excel format (.xlsx), reading a weighted or unweighted adjacency matrix where the regulators are in columns and the target genes are in rows, and automatically lays out and displays unweighted and weighted network graphs in a way that is familiar to biologists. Although GRNsight was originally developed for use with the GRNmap modeling software, and has provided useful insight in the interpretation of the gene regulatory network model described in Dahlquist et al. (2015), it has general applicability for displaying any small, unweighted or weighted network with directed edges for systems biology or other application domains. Thus, GRNsight inhabits a niche not satisfied by other software, doing "one thing well". GRNsight also serves as a model for how software engineering best practices can be learned simultaneously with the development of useful bioinformatics software.

Acknowledgments

We would like to thank Katrina Sherbina and B.J. Johnson for their input during the early stages of GRNsight development. We would also like to thank Masao Kitamura for assistance with setting up and administering the GRNsight server. Finally, we would like to thank the 2015-2016 GRNmap research team, Chukwuemeka E. Azinge, Juan S. Carrillo, Kristen M. Horstmann, Kayla C. Jackson, K. Grace Johnson, Brandon J. Klein, Tessa A. Morris, Margaret J. O'Neil, Trixie Anne M. Roque, and Natalie E. Williams, and the students enrolled in the Loyola Marymount University Spring 2015 course Biology 398-04: Biomathematical Modeling/Mathematics 388-01: Survey of Biomathematics for testing the software.

References

Bastian M., Heymann S., Jacomy M. 2009. Gephi: an open source software for exploring and manipulating networks. *Third International AAAI Conference on Weblogs and Social Media* 8:361–362.

Bostock M., Ogievetsky V., Heer J. 2011. D^3: Data-Driven Documents. *IEEE transactions on visualization and computer graphics* 17:2301–2309. DOI: 10.1109/TVCG.2011.185.

Brown E. 2014. *Web development with Node and Express*. Beijing ; Sebastopol, CA: O'Reilly. ISBN: 978-1-4919-4930-6

Dahlquist KD., Fitzpatrick BG., Camacho ET., Entzminger SD., Wanner NC. 2015. Parameter Estimation for Gene Regulatory Networks from Microarray Data: Cold Shock Response in Saccharomyces cerevisiae. *Bulletin of Mathematical Biology* 77:1457–1492. DOI: 10.1007/s11538-015-0092-6.

Dionisio JDN., Dahlquist KD. 2008. Improving the computer science in bioinformatics through

406 open source pedagogy. *ACM SIGCSE Bulletin* 40:115. DOI: 10.1145/1383602.1383648.
407 Franz M., Lopes CT., Huck G., Dong Y., Sumer O., Bader GD. 2016. Cytoscape.js: a graph
408 theory library for visualisation and analysis. *Bioinformatics (Oxford, England)* 32:309–311.
409 DOI: 10.1093/bioinformatics/btv557.
410 Gostner R., Baldacci B., Morine MJ., Priami C. 2014. Graphical Modeling Tools for Systems
411 Biology. *ACM Computing Surveys* 47:1–21. DOI: 10.1145/2633461.
412 Harbison CT., Gordon DB., Lee TI., Rinaldi NJ., Macisaac KD., Danford TW., Hannett NM.,
413 Tagne J-B., Reynolds DB., Yoo J., Jennings EG., Zeitlinger J., Pokholok DK., Kellis M.,
414 Rolfe PA., Takusagawa KT., Lander ES., Gifford DK., Fraenkel E., Young RA. 2004.
415 Transcriptional regulatory code of a eukaryotic genome. *Nature* 431:99–104. DOI:
416 10.1038/nature02800.
417 Lawlor B., Walsh P. 2015. Engineering bioinformatics: building reliability, performance and
418 productivity into bioinformatics software. *Bioengineered* 6:193–203. DOI:
419 10.1080/21655979.2015.1050162.
420 Lee TI., Rinaldi NJ., Robert F., Odom DT., Bar-Joseph Z., Gerber GK., Hannett NM., Harbison
421 CT., Thompson CM., Simon I., Zeitlinger J., Jennings EG., Murray HL., Gordon DB., Ren
422 B., Wyrick JJ., Tagne J-B., Volkert TL., Fraenkel E., Gifford DK., Young RA. 2002.
423 Transcriptional regulatory networks in Saccharomyces cerevisiae. *Science (New York, N.Y.)*
424 298:799–804. DOI: 10.1126/science.1075090.
425 Martin RC. (ed.) 2008. *Clean code: a handbook of agile software craftsmanship*. Upper Saddle
426 River, NJ: Prentice Hall. ISBN: 978-0-13-235088-4
427 Nielsen J. 1993. *Usability engineering*. Boston: Academic Press. ISBN: 978-0-12-518405-2

428 Norman DA. 2013. *The design of everyday things*. New York, New York: Basic Books. ISBN:
429 978-0-465-05065-9
430 Pavlopoulos GA., Malliarakis D., Papanikolaou N., Theodosiou T., Enright AJ., Iliopoulos I.
431 2015. Visualizing genome and systems biology: technologies, tools, implementation
432 techniques and trends, past, present and future. *GigaScience* 4:38. DOI: 10.1186/s13742-015-
433 0077-2.
434 Salomonis N., Hanspers K., Zambon AC., Vranizan K., Lawlor SC., Dahlquist KD., Doniger
435 SW., Stuart J., Conklin BR., Pico AR. 2007. GenMAPP 2: new features and resources for
436 pathway analysis. *BMC bioinformatics* 8:217. DOI: 10.1186/1471-2105-8-217.
437 Schade B., Jansen G., Whiteway M., Entian KD., Thomas DY. 2004. Cold adaptation in budding
438 yeast. *Molecular Biology of the Cell* 15:5492–5502. DOI: 10.1091/mbc.E04-03-0167.
439 Schultheiss SJ., Münch M-C., Andreeva GD., Rätsch G. 2011. Persistence and availability of
440 Web services in computational biology. *PloS One* 6:e24914. DOI:
441 10.1371/journal.pone.0024914.
442 Schultheiss SJ. 2011. Ten simple rules for providing a scientific Web resource. *PLoS*
443 *computational biology* 7:e1001126. DOI: 10.1371/journal.pcbi.1001126.
444 Shannon P., Markiel A., Ozier O., Baliga NS., Wang JT., Ramage D., Amin N., Schwikowski B.,
445 Ideker T. 2003. Cytoscape: a software environment for integrated models of biomolecular
446 interaction networks. *Genome Research* 13:2498–2504. DOI: 10.1101/gr.1239303.
447 Shneiderman B., Plaisant C., Cohen M., Jacobs SM., Elmqvist N., Diakopoulos N. 2016.
448 *Designing the user interface: strategies for effective human-computer interaction*. Hoboken:
449 Pearson. ISBN: 978-0-13-438038-4
450 Shore D., Nasmyth K. 1987. Purification and cloning of a DNA binding protein from yeast that

binds to both silencer and activator elements. *Cell* 51:721–732.

Smoot ME., Ono K., Ruscheinski J., Wang P-L., Ideker T. 2011. Cytoscape 2.8: new features for data integration and network visualization. *Bioinformatics (Oxford, England)* 27:431–432. DOI: 10.1093/bioinformatics/btq675.

Teixeira MC., Monteiro PT., Guerreiro JF., Gonçalves JP., Mira NP., dos Santos SC., Cabrito TR., Palma M., Costa C., Francisco AP., Madeira SC., Oliveira AL., Freitas AT., Sá-Correia I. 2014. The YEASTRACT database: an upgraded information system for the analysis of gene and genomic transcription regulation in Saccharomyces cerevisiae. *Nucleic Acids Research* 42:D161–166. DOI: 10.1093/nar/gkt1015.

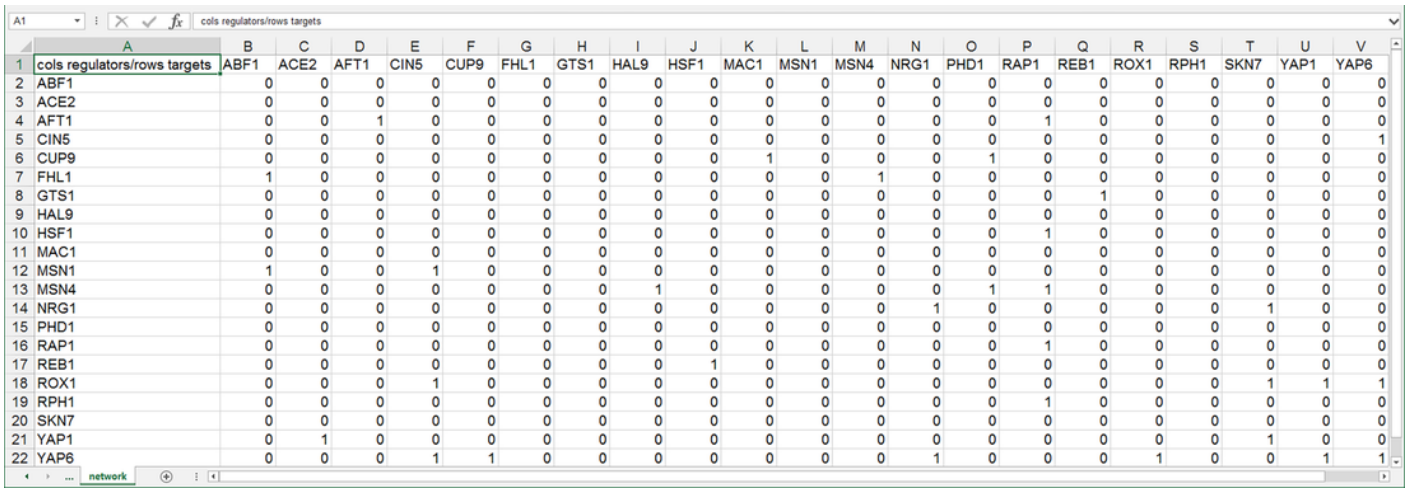
Veretnik S., Fink JL., Bourne PE. 2008. Computational biology resources lack persistence and usability. *PLoS computational biology* 4:e1000136. DOI: 10.1371/journal.pcbi.1000136.

Wilson G., Aruliah DA., Brown CT., Chue Hong NP., Davis M., Guy RT., Haddock SHD., Huff KD., Mitchell IM., Plumbley MD., Waugh B., White EP., Wilson P. 2014. Best practices for scientific computing. *PLoS biology* 12:e1001745. DOI: 10.1371/journal.pbio.1001745.

1

Screenshot of the expected format for an adjacency matrix for an unweighted network.

Regulators are named in the columns and target genes in the rows. A gene name at the top of the matrix will be considered the same as a gene name on the side if it contains the same text string, regardless of capitalization.

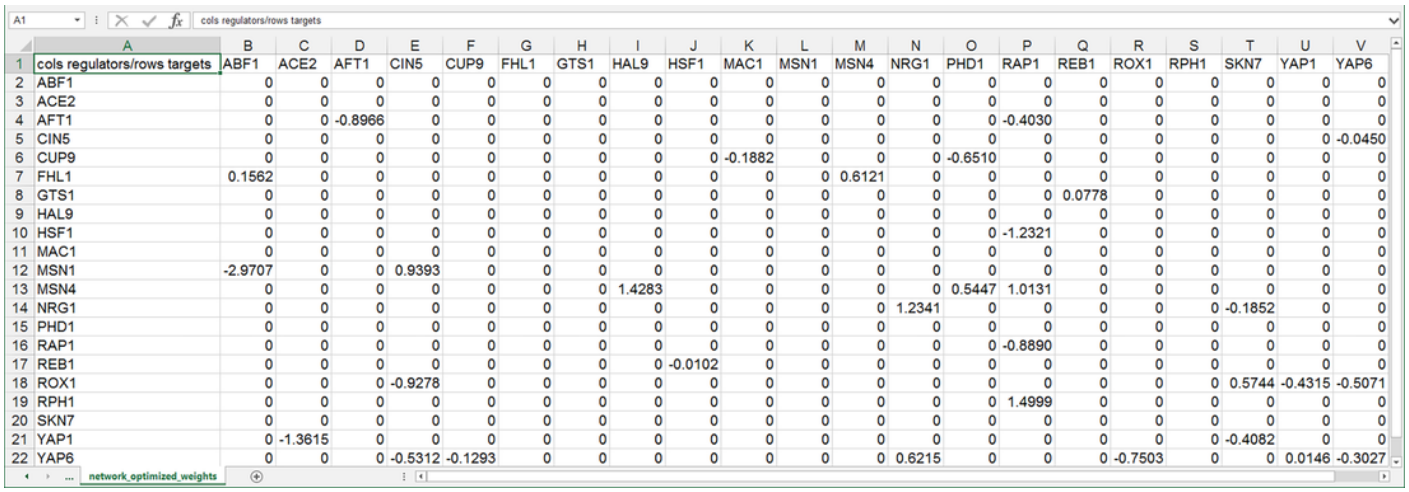


cols regulators/rows targets	ABF1	ACE2	AFT1	CIN5	CUP9	FHL1	GTS1	HAL9	HSF1	MAC1	MSN1	MSN4	NRG1	PHD1	RAP1	REB1	ROX1	RPH1	SKN7	YAP1	YAP6
ABF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ACE2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AFT1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
CIN5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
CUP9	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
FHL1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
GTS1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
HAL9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HSF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
MAC1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MSN1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MSN4	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0
NRG1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0
PHD1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RAP1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
REB1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
ROX1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
RPH1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
SKN7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YAP1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
YAP6	0	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1	1

2

Screenshot of the expected format for an adjacency matrix for a weighted network.

Regulators are named in the columns and target genes in the rows. A gene name at the top of the matrix will be considered the same as a gene name on the side if it contains the same text string, regardless of capitalization.



cols regulators/rows targets	ABF1	ACE2	AFT1	CIN5	CUP9	FHL1	GTS1	HAL9	HSF1	MAC1	MSN1	MSN4	NRG1	PHD1	RAP1	REB1	ROX1	RPH1	SKN7	YAP1	YAP6
ABF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ACE2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
AFT1	0	0	-0.8966	0	0	0	0	0	0	0	0	0	0	0	-0.4030	0	0	0	0	0	0
CIN5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.0450
CUP9	0	0	0	0	0	0	0	0	0	-0.1882	0	0	0	-0.6510	0	0	0	0	0	0	0
FHL1	0.1562	0	0	0	0	0	0	0	0	0	0	0.6121	0	0	0	0	0	0	0	0	0
GTS1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.0778	0	0	0	0	0
HAL9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
HSF1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1.2321	0	0	0	0	0	0
MAC1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MSN1	-2.9707	0	0	0.9393	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
MSN4	0	0	0	0	0	0	0	1.4283	0	0	0	0	0	0.5447	1.0131	0	0	0	0	0	0
NRG1	0	0	0	0	0	0	0	0	0	0	0	0	1.2341	0	0	0	0	0	-0.1852	0	0
PHD1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RAP1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.8890	0	0	0	0	0	0
REB1	0	0	0	0	0	0	0	0	-0.0102	0	0	0	0	0	0	0	0	0	0	0	0
ROX1	0	0	0	-0.9278	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.5744	-0.4315	-0.5071
RPH1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.4999	0	0	0	0	0	0
SKN7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YAP1	0	-1.3615	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-0.4082	0	0
YAP6	0	0	0	-0.5312	-0.1293	0	0	0	0	0	0	0	0.6215	0	0	0	-0.7503	0	0	0.0146	-0.3027

Figure 3(on next page)

GRNsight architecture and component interactions.

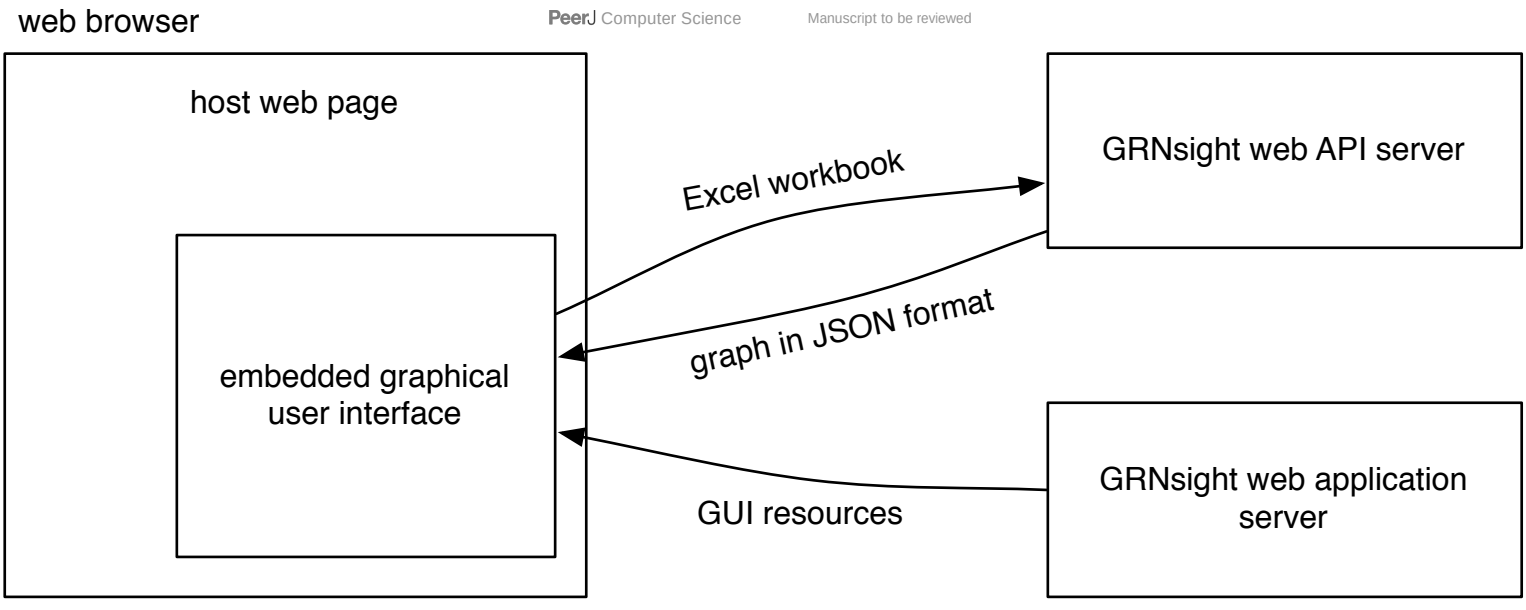

web browser
host web page
embedded graphical user interface
Excel workbook
graph in JSON format
GUI resources
GRNsight web API server
GRNsight web application server

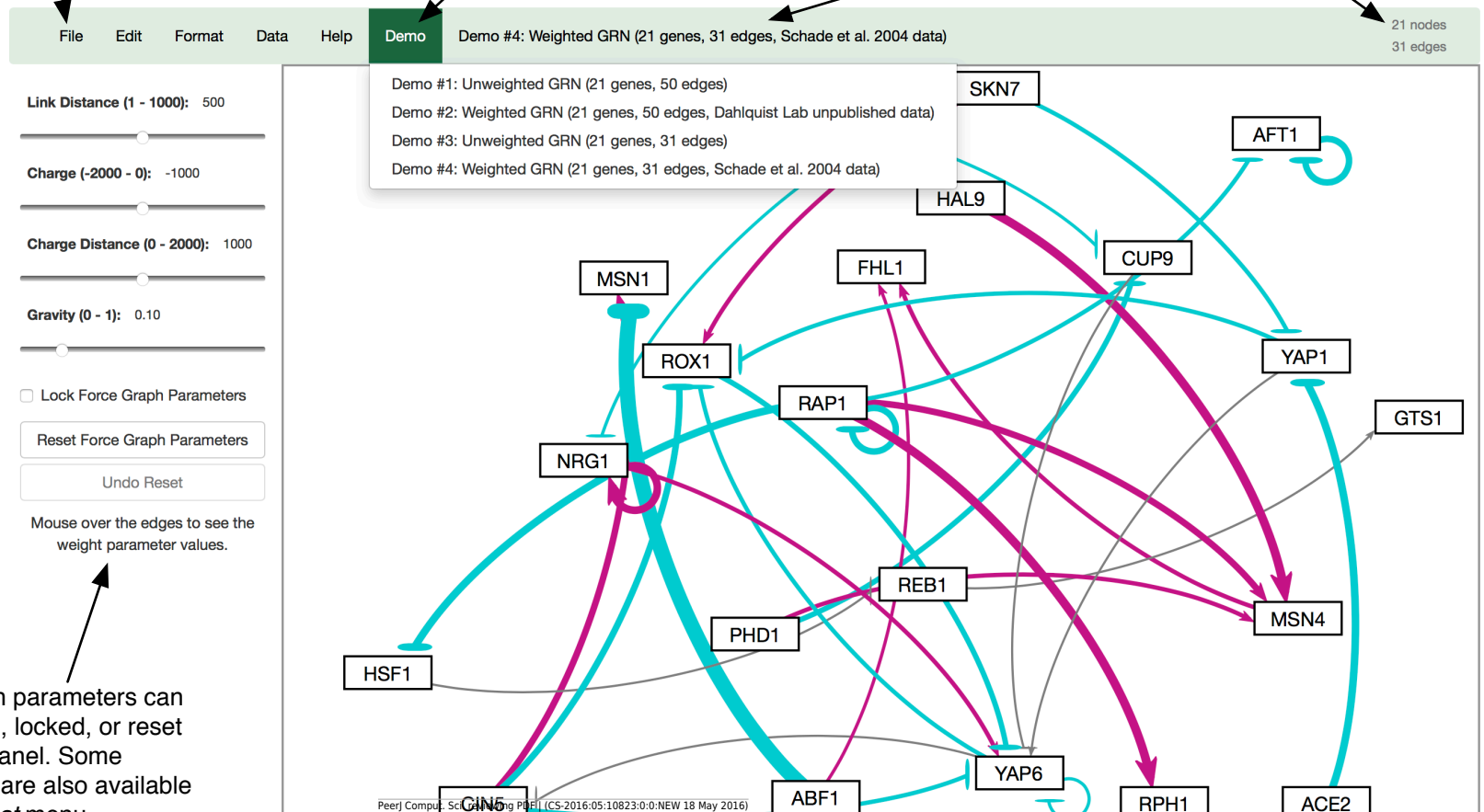
Figure 4(on next page)

Annotated screenshot of the GRNsight user interface.

The File menu includes commands for uploading an adjacency matrix in Microsoft Excel (.xlsx) format.

The Demo menu lists four GRNs that have been preloaded into the server.

The status display shows the current demo or filename along with node and edge counts.



Force graph parameters can be adjusted, locked, or reset using this panel. Some commands are also available in the *Format* menu.

Table 1(on next page)

GRNsight test suite code coverage summary.

Denominators represent the number of aspects of each type detected by Istanbul in the GRNsight codebase; numerators represent the subset of these which were executed by unit test code.

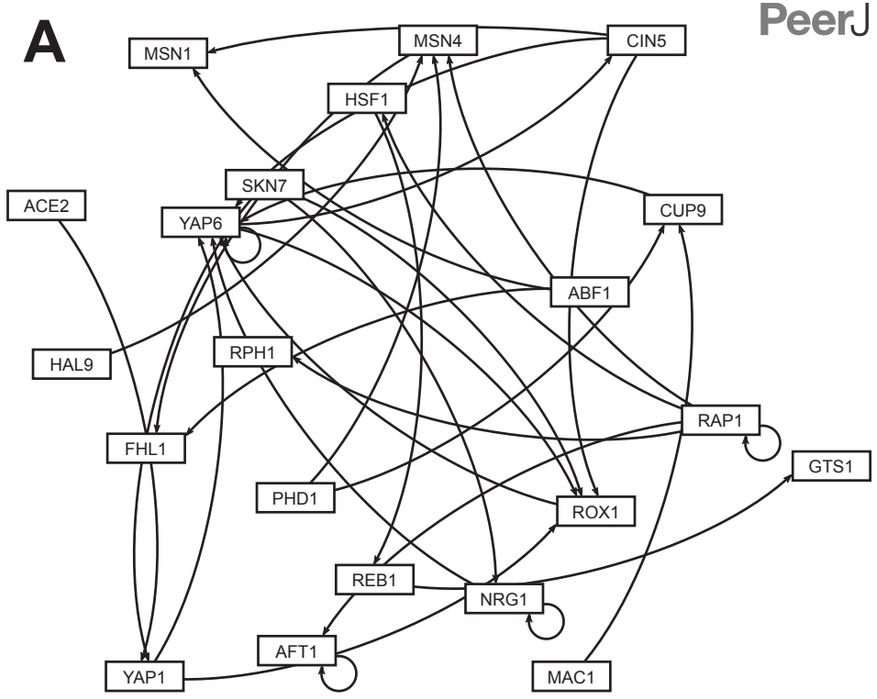
Aspect of the Code	Test Coverage (percent)
Statements	112/162 (69.1%)
Branches	59/68 (86.8%)
Functions	15/24 (62.5%)
Lines	112/157 (71.3%)

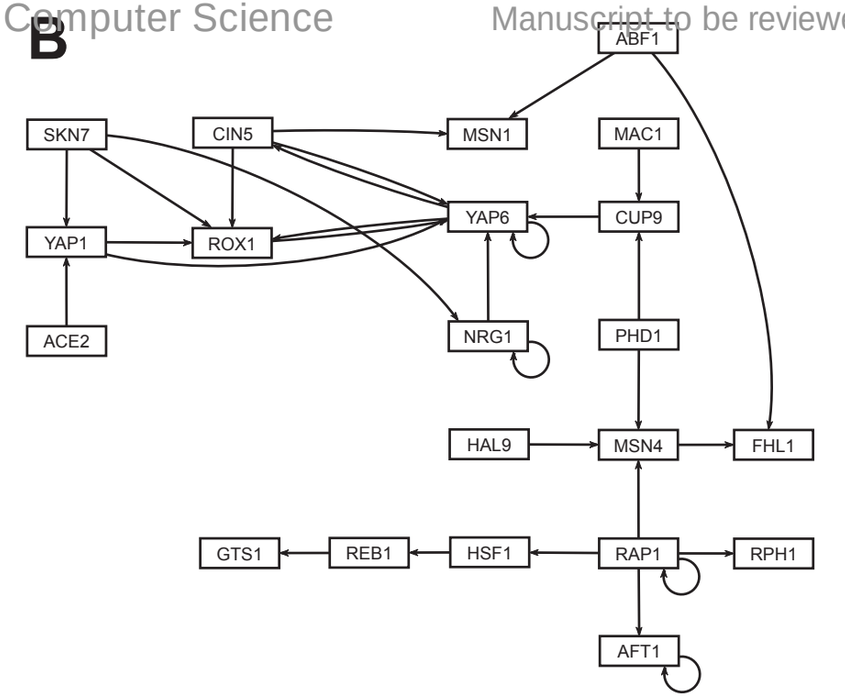
1

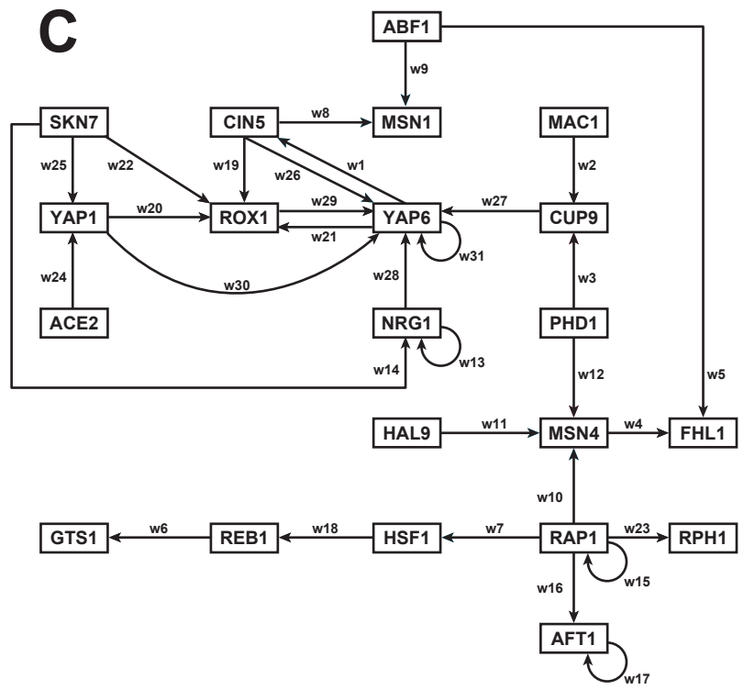
Figure 5(on next page)

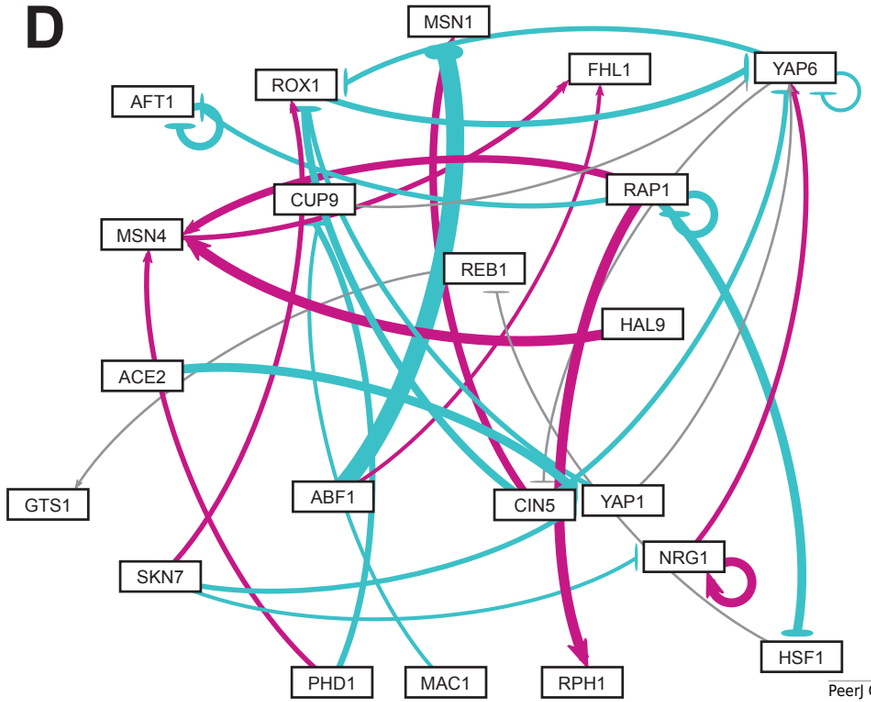
Side-by-side comparison of the same adjacency matrices laid out by GRNsight and by hand.

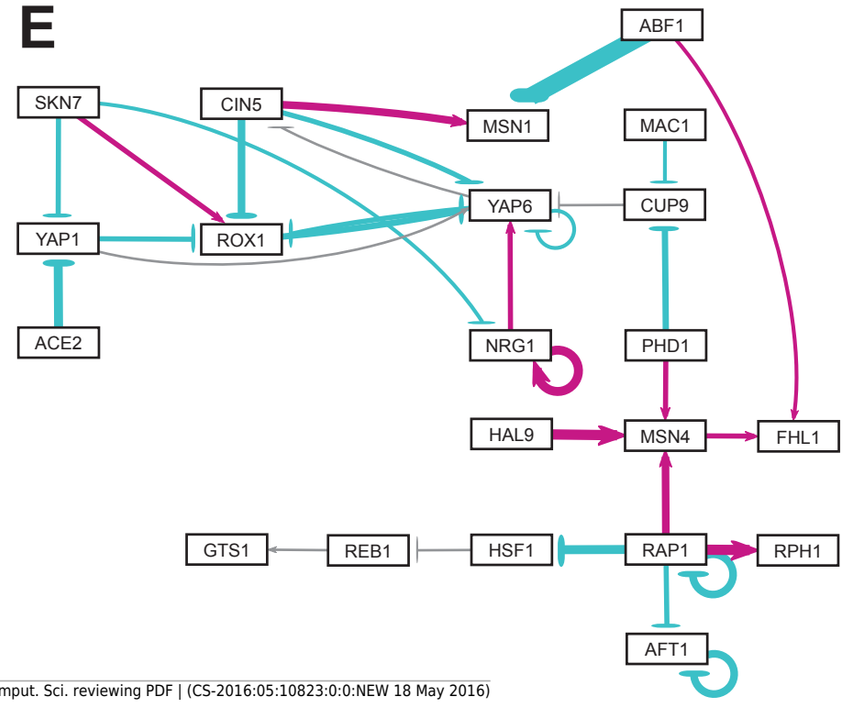
A) GRNsight automatic layout of the demonstration file, Demo #3: Unweighted GRN (21 genes, 31 edges); B) graph from (A) manually manipulated from within GRNsight; C) the same adjacency matrix from (A) and (B) laid out entirely by hand in Adobe Illustrator, corresponding to Figure 1 of Dahlquist et al., (2015); D) GRNsight automatic layout of the demonstration file, Demo #4: Weighted GRN (21 genes, 31 edges, Schade et al. 2004 data); E) graph from (D) manually manipulated from within GRNsight; F) the same adjacency matrix from (D) and (E) laid out entirely by hand in Adobe Illustrator, corresponding to Figure 8 of Dahlquist et al., (2015).

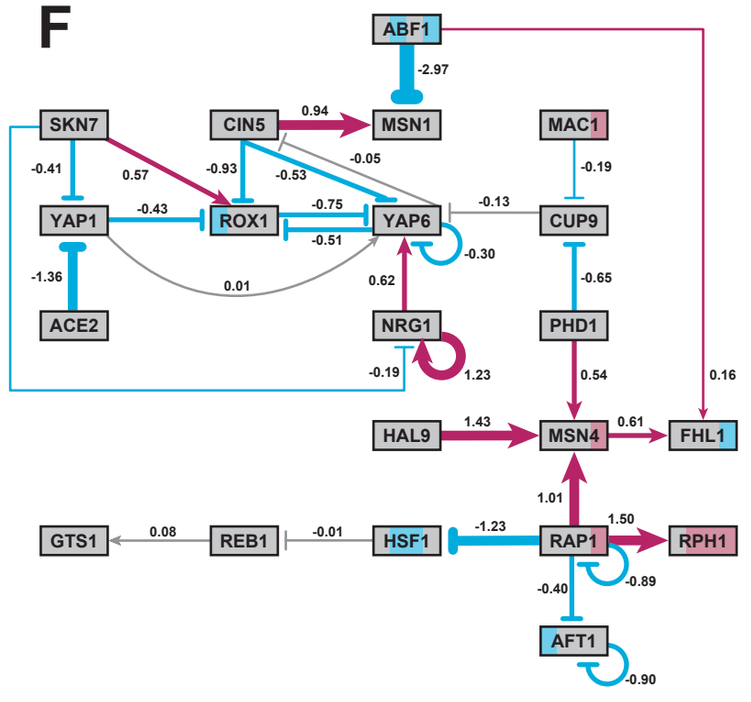

A
MSN1
MSN4
CIN5
HSF1
SKN7
ACE2
YAP6
CUP9
ABF1
HAL9
RPH1
RAP1
FHL1
GTS1
PHD1
ROX1
REB1
NRG1
AFT1
YAP1
MAC1


B
ABF1
SKN7
CIN5
MSN1
MAC1
YAP1
ROX1
YAP6
CUP9
ACE2
NRG1
PHD1
HAL9
MSN4
FHL1
GTS1
REB1
HSF1
RAP1
RPH1
AFT1


C
ABF1
w9
SKN7
CIN5
w8
MSN1
MAC1
w25
w22
w19
w26
w1
w2
YAP1
w20
ROX1
w29
YAP6
w27
CUP9
w21
w31
w24
w30
w28
w3
ACE2
NRG1
PHD1
w14
w13
w12
w5
HAL9
w11
MSN4
w4
FHL1
w10
GTS1
w6
REB1
w18
HSF1
w7
RAP1
w23
RPH1
w16
w15
AFT1
w17


D
MSN1
FHL1
YAP6
AFT1
ROX1
RAP1
CUP9
MSN4
REB1
HAL9
ACE2
GTS1
ABF1
CIN5
YAP1
NRG1
SKN7
HSF1
PHD1
MAC1
RPH1


E
ABF1
SKN7
CIN5
MSN1
MAC1
YAP1
ROX1
YAP6
CUP9
ACE2
NRG1
PHD1
HAL9
MSN4
FHL1
GTS1
REB1
HSF1
RAP1
RPH1
AFT1


F
ABF1
-2.97
SKN7
CIN5
0.94
MSN1
MAC1
-0.41
0.57
-0.93
-0.53
-0.05
-0.19
YAP1
-0.43
ROX1
-0.75
YAP6
-0.13
CUP9
-0.51
-0.30
-1.36
0.01
0.62
-0.65
ACE2
NRG1
PHD1
-0.19
1.23
0.54
0.16
HAL9
1.43
MSN4
0.61
FHL1
1.01
GTS1
0.08
REB1
-0.01
HSF1
-1.23
RAP1
1.50
RPH1
-0.40
-0.89
AFT1
-0.90