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

Kam D. Dahlquist1\*, John David N. Dionisio2, Ben G. Fitzpatrick3, Nicole A. Anguiano2, Anindita Varshneya1, Britain J. Southwick2, Mihir Samdarshi1

1Department of Biology, 2Department of Electrical Engineering and Computer Science, 3Department 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 around 160 automated unit tests that examine nearly 530 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/>. [add link to repository; add import/export to SIF/GraphML; requires some editing down because (I think) hard limit on 3000 characters]

# 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 unweighted and weighted network graphs corresponding to the matrix without having to create or modify MATLAB code. Given that our user base included students in courses using university computer labs where the installation and maintenance of software is subject to logistical considerations sometimes beyond our control, we enumerated the following requirements for a potential visualization tool. The tool should:

1. Exist as a web application without the need to download and install specialized software;
2. Be simple and intuitive to use;
3. Accept an input file in Microsoft Excel format (.xlsx);
4. Read a weighted or unweighted adjacency matrix where the regulatory transcription factors are in columns and the target genes are in rows;
5. Automatically lay out and display small- to medium-scale, unweighted and weighted, directed network graphs in a way that is familiar to biologists and adds value to the interpretation of the modeling results.

Having established the broad technical requirements to which we were seeking a solution, the first task was to determine if software already existed that could fulfill our needs. A review by Pavlopoulos et al., published in 2015, describes the types, trends, and usage of visualization tools available for genomics and systems biology. Their list of 47 tools for network analysis is representative of what was available to us at our project inception in January 2014 (given the caveat that the list itself is a moving target with some tools dropping out, new ones being added, and others evolving in their functions). With such a large number of tools available, it would be reasonable to expect that one already existed that could fulfill our needs. However, our use case was narrow, and the tools we investigated out of this diverse set each had properties that limited their use for us. With regard to our first requirement, out of the 47 tools, 29 are stand-alone applications, requiring installation, versus 18 web applications. With respect to our second requirement, the more complex software packages out of the set have a steep learning curve. Our third and fourth requirements specify datatypes. Some packages were hard coded for a different type of network than a GRN (e.g., metabolic or signaling pathways, protein-protein interaction networks) or retrieved data exclusively from a backend database, not allowing user-supplied data. None at the time would readily accept an adjacency matrix with the GRNmap specifications as input without some manipulation of the data format. Finally, with respect to the last requirement, the core functionality, some packages were designed for visualization and analysis of much larger networks than the ones in which we were interested or did not have the ability to display directed, weighted graphs.

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.

After this exploration, we decided to create our own software solution that we could exactly tailor to our specifications. Following the philosophy of “do one thing well” (*http://onethingwell.org/post/457050307/about-one-thing-well*), we wanted to prioritize rendering 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. Similarly, we wanted to pass data seamlessly from GRNmap to GRNsight, while bearing in mind that we should adopt practices that would make our tool also useful to users outside our own group. Finally, we wanted to minimize any startup, onboarding, or overhead time, which necessitated also enumerating a set of process requirements that would lead us to our goal:

* Follow best practices for open software development in bioinformatics, including: reusing code, releasing early and often to a public repository, tracking requirements, issues, and bugs, performing unit-tests, and providing both code and user documentation (Schultheiss, 2011; Prlic and Procter; Wilson et al., 2014).
* Leverage the expertise of the faculty and undergraduate student development team members and be responsive to our GRNmap customers (i.e., eat our own dog food).
* Balance the needs of fulfilling our own use case with developing a tool of wider applicability to the scientific community during a development cycle that ebbs and flows with the pressures of the academic calendar.

GRNsight both fulfills our stated product requirements and serves as a model for best practices for software development in bioinformatics as discussed in the sections 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.

After having implemented the core functionality of seamlessly reading GRNmap-generated Excel workbooks, we recently extended the ability of GRNsight to read other commonly used network data formats to increase the interoperability of GRNsight with other network analysis and visualization software. GRNsight can import and display Simple Interaction Format (SIF, .sif, *http://manual.cytoscape.org/en/latest/Supported\_Network\_File\_Formats.html#sif-format*) and GraphML (.graphml, Brandes et al. 2001, *http://graphml.graphdrawing.org/*) files and export network data in those two formats (See the GRNsight documentation page for details of the implementation at *http://dondi.github.io/GRNsight/documentation.html.*)

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 around 160 automated unit tests that examine nearly 530 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. The web site is free and open to all users, and there is no login requirement. 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 12 August 2016, the GRNsight home page has been accessed 2349 times, and 1652 files have been uploaded and viewed with GRNsight. Of these 1652 files, an estimated 148 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 or ACE2. 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 edge corresponding to the repression of the expression of MSN1 by ABF1 stands out as the thickest because the absolute value of its weight parameter (-2.97) has the largest magnitude out of all the weights (Dahlquist et al., 2015). It is noticeable that none of the edges that represent activation are as thick as the ABF1-to-MSN1 edge; only RAP1-to-RPH1 and HAL9-to-MSN4 are close with weights of 1.50 and 1.43, respectively.

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). Taking the arrowhead type, thickness, and color into consideration, one can, by visual inspection, group edges by type and relative influence into four activation and four repression bins. RAP1-to-RPH1, HAL9-to-MSN4, and NRG1 to itself have the strongest activation relationships, followed by CIN5-to-MSN1, followed by NRG1-to-YAP6, MSN4-to-FHL1, SKN7-to ROX1 and PHD1-to-MSN4, followed by ABF1-to-FHL1 as the weakest of the activation relationships. The aforementioned ABF1-to-MSN1 edge has the strongest repression relationship, followed by ACE2-to-YAP1, RAP1-to-HSF1, CIN5-to-ROX1, AFT1 to itself, and RAP1 to itself, followed by ROX1-to-YAP6, PHD1-to-CUP9, CIN5-to-YAP6, YAP6-to-ROX1, YAP1-to-ROX1, SKN7-to-YAP1, RAP1-to-AFT1, and YAP6 to itself, followed by MAC1-to-CUP9 and SKN7-to-NRG1 as the weakest of the repression relationships. These rankings could have been obtained, of course, by sorting the numerical values of the edges in a table, but it is noteable that these groupings can also be picked out by eye and then put into the context of the other network connections.

Because the five weakest connections, CUP9-to-YAP6, REB1-to-GTS1, YAP6-to-CIN5, YAP1-to-YAP6, and HSF1-to-REB1, colored grey, are de-emphasized in the visual display, a different interpretation of the network structure can be made as compared to the unweighted network (Fig. 5E and F versus 5B and C). In most cases, nodes in a regulatory chain “drop out” visually “breaking” the chain. For example in the four-node chain beginning with RAP1-to-HSF1, the last two nodes, REB1 and GTS1, are only weakly connected. In the five-node chains beginning with SKN7-to-YAP1 or ACE2-to-YAP1, and the four-node chains beginning with MAC1-to-CUP9 or PHD1-to-CUP9, the nodes connected to YAP6 drop out (YAP1-to-YAP6, YAP6-to-CIN5, and CUP9-YAP6). This suggests that regulatory chains may only be effective to a depth of two levels, and that while longer chains are theoretically possible, given the network connections, they have a negligible effect on the dynamics of expression of downstream genes.

Another interpretation of the network structure that is highlighted by the weighted display is that the 21-gene network could be divided into two smaller subnetworks by removing the two edges CUP9-to-YAP6 (grey) and ABF1-to-FHL1 (thin magenta, weakly activating). Finally, the unweighted display showed two complex feedforward motifs involving CIN5, ROX1, and YAP6 and SKN7, YAP1, and ROX1. The weighted display reveals that the complexity of the connections is reduced because the weak YAP1-to-YAP6 and YAP6-to-CIN5 edges drop out. Furthermore, the display shows that the modeling predicts that the three-node CIN5-ROX1-YAP6 motif is an incoherent type 2 feedforward loop, while the SKN7-YAP1-ROX1 motif is a coherent type 4 feedforward loop, neither of which is found very commonly in *Escherichia coli* or *S. cerevisiae* gene regulatory networks (Alon 2007). The modeling combined with the display suggests that further investigation is warranted: either these two rare types of feedforward loops are important to the dynamics of this particular GRN, or the network structure is incorrect. In either case, future lines of experimental investigation are suggested to the user.

When examining individual genes in the network, one can see that 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. The expression of ROX1 is weakly activated by SKN7 and weakly repressed by YAP1, CIN5, and YAP6. The expression of YAP6 is weakly activated by NRG1, but weakly repressed by itself, CIN5, and ROX1. 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. PHD1, ABF1, CIN5, and SKN7 also both activate and repress their different target genes in the network. For each of these regulators, there is experimental evidence to support their opposite effects on gene expression, although not necessarily for these particular target gnees (RAP1: Shore and Nasmyth, 1987; PHD1: Borneman et al. 2006, ABF1: Buchman and Kornberg 1990, Miyake et al. 2004; CIN5 and SKN7: Nie et al. 2009). Except for CIN5, what these genes have in common is that they themselves have no inputs in the network. The remaining no-input genes (ACE2, MAC1, and HAL9) have only one outgoing edge in this network. Because these genes have no imputs and, in some sense, have been artifically disconnected from the larger GRN in the cell, one must not overinterpret the results of the modeling for these genes.

Thus, GRNsight enables one to interpret the weight parameters more easily than one could from the adjacency matrix alone. Visual inspection has long been recognized by experts such as Tufte (1983) and Card, Mackinlay, and Shneiderman (1999) as distinct from other forms of purely computational or algorithmic data analysis, and as the preceding discussion highlights, it is this potential that can be derived specifically by visual inspection that is enabled by GRNsight. Card, Mackinlay, and Shneiderman (1999) have identified six major ways, documented in earlier literature and empirical studies, by which information visualization amplifies cognition. Tufte’s seminal book *The Visual Display of Quantitative Information* (1983) perhaps states and demonstrates it best: “Graphics reveal data. Indeed graphics can be more precise and revealing than conventional statistical computations.”

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 any particular species, nor 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

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.

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).

## GRNsight Complies with FAIR Data Principles [need better verb]

The FAIR Guiding Principles for scientific data and stewardship state that data should be Findable, Accessible, Interoperable, and Reusable by both humans and machines (Wilkinson et al. 2016), with “data” loosely construed as any scholarly digital research object, including software. As scientific software that interacts with data, the FAIR principles can apply to both the GRNsight application and the network data it is used to visualize. Thus, we evaluate the GRNsight project in terms of its “FAIRness” below.

### Findable

We have made GRNsight Findable by registering it with well-known bioinformatics tools registries: the BioJS Repository (Yachdav et al. 2015, http://biojs.io/), the Elixir Tools and Data Services Registry (Ison et al. 2016, https://bio.tools/), Bioinformatics.org (http://www.bioinformatics.org/wiki/), and the Links Directory at Bioinformatics.ca (Brazas, Yamada & Ouellette 2010), https://bioinformatics.ca/links\_directory/), as well as NPM (Node Package Manager, https://www.npmjs.com/). GRNsight has been presented at scientific conferences, with slides and posters available via SlideShare (http://www.slideshare.net/GRNsight) and with a recent talk and poster at the 2016 Bioinformatics Open Source Conference available via F1000 Research (Dahlquist KD, Fitzpatrick BG, Dionisio JDN et al. GRNmap and GRNsight: open source software for dynamical systems modeling and visualization of medium-scale gene regulatory networks [v1; not peer reviewed]. *F1000Research* 2016, **5**(ISCB Comm J):1637 (slides) (doi: [10.7490/f1000research.1112534.1](http://dx.doi.org/10.7490/f1000research.1112534.1) and Dahlquist KD, Fitzpatrick BG, Dionisio JDN et al. GRNmap and GRNsight: open source software for dynamical systems modeling and visualization of medium-scale gene regulatory networks [v1; not peer reviewed]. *F1000Research* 2016, **5**(ISCB Comm J):1618 (poster) (doi: [10.7490/f1000research.1112518.1](http://dx.doi.org/10.7490/f1000research.1112518.1))). We have paid special care to the meta data associated with our web site to increase its Findability via Google search. And of course, with the publication of this article, GRNsight is Findable in literature databases. With the inclusion of the site and repository URL in the abstract, the code is findable by automated text mining [ref]. The Findable guiding principle states that meta data and data should have a globally unique and persistent identifier (Wilkinson et al. 2016). In terms of software, this would be the version. Because we utilize the GitHub release mechanism, code is tagged with a version and each version is available from the release page (https://github.com/dondi/GRNsight/releases). Because GRNsight does not interact directly with a data repository, it is up to individual users to make sure that their data is FAIR compliant with the Findable principle. One could argue that by being “yet another” network visualization tool in a crowded domain (recall 47 other tools recorded by Pavlopoulos et al. 2015), GRNsight is contributing to a Findability problem for users in the sense that it contributes more “hay” to the needle in a haystack problem of finding the right tool for the job. However, we hope that by the actions we have taken, the benefits of adding GRNsight to the diverse pool of network visualization software outweighs the detriments.

### Accessible

As noted in the section on Availability, the GRNsight web application is accessible because it is free and open to all users, and there is no login requirement. The source code is available under the open source BSD license and can be npm installed (given the caveat that the user must be able to support the GRNsight client-server setup). The longevity of GRNsight is partially tied to the longevity of the GitHub repository itself, although the authors maintain local backups. Again, because GRNsight does not interact directly with a data repository, it is up to individual users to make sure that their data is FAIR compliant with the Accessible principle. Since GRNsight does not have any security [authentication?] procedures (e.g., login for registered users), it is not recommended that sensitive data be uploaded to our GRNsight server. However, users who wish to visualize sensitive data could run a local instance of the GRNsight client-server setup.

### Interoperable

As software, GRNsight is *not* Interoperable in the sense that it interacts directly with other software or databases, as, for example, Cytoscape does with xx database or individual Cytoscape apps (formerly plugins) [ref]. However, GRNsight *is* Interoperable in the sense that it can receive and pass data from and to other programs. In this latter sense, this section could just as easily have been entitled, “95% of bioinformatics is getting your data into the right file format.” Indeed, one of the original motivations and requirements for GRNsight was to seamlessly read and display weighted GRNs that were output as Excel workbooks from the GRNmap modeling MATLAB package (Dahlquist et al. 2015, http://kdahlquist.github.io/GRNmap/). However, as this is a specialized use case, we have recently implemented the ability for GRNsight to import and export data in SIF (*http://manual.cytoscape.org/en/stable/Supported\_Network\_File\_Formats.html#sif-format*) and GraphML format (*http://graphml.graphdrawing.org/*, Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M., & Marshall, M. S. (2001, September). GraphML progress report structural layer proposal. In *International Symposium on Graph Drawing* (pp. 501-512). Springer Berlin Heidelberg), facilitating movement of data between GRNsight and other network visualization and analysis programs. Thus, we are in a position to comment on these two formats with respect to the finer points of data Interoperability, including 1) metadata and data use a formal, accessible, shared, and broadly applicable language for knowledge representation; 2) metadata and data use vocabularies that follow the FAIR principles; and 3) metadata and data include qualified references to other metadata and data (quoted from Box 2 of Wilkinson et al. 2016).

When we implemented import and export for the SIF and GraphML formats, we encountered issues due to the flexibility with which the formats are specified that required design decisions that may, in turn, restrict compatibility with software with which we did not test. For example, the SIF format as described in the documentation for Cytoscape 3.4.0, offers quite a bit of flexibility, including choice of delimiter (space vs. tab), denoting a pairwise list of interactions versus concatenating all the interactions to the same node on the same line, and the choice of relationship type (any string). While GRNsight strives to read any SIF file, we restricted our export format to tab-delimited, pairwise interactions, and a single relationship type (“pd” for “protein 🡪 DNA”) for unweighted networks. For weighted networks, GRNsight exports the weight value as the relationship type. The advantage of SIF is that it is a simple text format; the main disadvantage is that all it is really intended to encode is the interaction between two nodes, which makes including the weight data as GRNsight does a kludge. As a simple text format, it does not satisfy the three sub-principles of Interoperability described above (Wilkinson et al. 2016). In particular, there is no controlled vocabulary for the relationship type, only a list of suggestions in the Cytoscape documentation, from which we selected “pd”. In practice, Cytoscape defaults to “interacts with” as the relationship type when exporting SIF files.

GraphML has an entry in the biosharing.org registry of standards (McQuilton, P., Gonzalez-Beltran, A., Rocca-Serra, P., Thurston, M., Lister, A., Maguire, E., & Sansone, S. A. (2016). BioSharing: curated and crowd-sourced metadata standards, databases and data policies in the life sciences. *Database*, *2016*, baw075.), but as of this writing is an unclaimed, automatically-generated entry. We are not using a bunch of options (nested networks, undirected networks), but encountered an issue with ID. In the GRNsight-native Excel format, transcription factors must be unique in the column and row and serve both as a unique ID for that node and the node label. In two implementations of GraphML import/export that we tested with Cytoscape and yED, the node ID was separate from the node label and not editable by the user. This leads to a situation where the same label could be assigned to nodes with different IDs and an issue for correct display of the network in GRNsight. We are not attempting to export any layout information, just data. Again, no controlled vocabulary, flexibility of defining key element allows for variety of solutions for encoding information. We made sure that we could read yED and Cytoscape-exported GraphML and that GRNsight-exported GraphML was accurately read by these two programs, but cannot guarantee what will happen with other software.

### Reusable

Finally, GRNsight is reusable because the code is avaialable on GitHub under the open source BSD license. The advantage of having followed test-driven development is that a developer who wishes to reuse the code has a test suite ready to guide development of new features. In terms of data, the criteria for reusability are closely linked to interoperability. Only the GraphML format is capable of storing metadata, but the limitations described above in terms of a lack of controlled vocabulary fail the Reusability test as well. In terms of provenance, we have injected a comment into the GraphML recording what version of GRNsight exported the data.

FAIR Guiding Principles have only recently been published, this may be the first discussion of how to explicity apply them to software. While GRNsight has limitations listed above, we have done as much as we can to achieve FAIRness at this time.

# 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. D3: 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

Card SK., Mackinlay JD., Shneiderman B. 1999. “Chapter 1: Information Visualization.” In *Readings in Information Visualization: Using Vision to Think*. San Diego, California: Academic Press. ISBN: 978-1-5586-0533-6

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 open source pedagogy. *ACM SIGCSE Bulletin* 40:115. DOI: 10.1145/1383602.1383648.

Franz M., Lopes CT., Huck G., Dong Y., Sumer O., Bader GD. 2016. Cytoscape.js: a graph theory library for visualisation and analysis. *Bioinformatics (Oxford, England)* 32:309–311. DOI: 10.1093/bioinformatics/btv557.

Gostner R., Baldacci B., Morine MJ., Priami C. 2014. Graphical Modeling Tools for Systems Biology. *ACM Computing Surveys* 47:1–21. DOI: 10.1145/2633461.

Harbison CT., Gordon DB., Lee TI., Rinaldi NJ., Macisaac KD., Danford TW., Hannett NM., Tagne J-B., Reynolds DB., Yoo J., Jennings EG., Zeitlinger J., Pokholok DK., Kellis M., Rolfe PA., Takusagawa KT., Lander ES., Gifford DK., Fraenkel E., Young RA. 2004. Transcriptional regulatory code of a eukaryotic genome. *Nature* 431:99–104. DOI: 10.1038/nature02800.

Lawlor B., Walsh P. 2015. Engineering bioinformatics: building reliability, performance and productivity into bioinformatics software. *Bioengineered* 6:193–203. DOI: 10.1080/21655979.2015.1050162.

Lee TI., Rinaldi NJ., Robert F., Odom DT., Bar-Joseph Z., Gerber GK., Hannett NM., Harbison CT., Thompson CM., Simon I., Zeitlinger J., Jennings EG., Murray HL., Gordon DB., Ren B., Wyrick JJ., Tagne J-B., Volkert TL., Fraenkel E., Gifford DK., Young RA. 2002. Transcriptional regulatory networks in Saccharomyces cerevisiae. *Science (New York, N.Y.)* 298:799–804. DOI: 10.1126/science.1075090.

Martin RC. (ed.) 2008. *Clean code: a handbook of agile software craftsmanship*. Upper Saddle River, NJ: Prentice Hall. ISBN: 978-0-13-235088-4

Nielsen J. 1993. *Usability engineering*. Boston: Academic Press. ISBN: 978-0-12-518405-2

Norman DA. 2013. *The design of everyday things*. New York, New York: Basic Books. ISBN: 978-0-465-05065-9

Pavlopoulos GA., Malliarakis D., Papanikolaou N., Theodosiou T., Enright AJ., Iliopoulos I. 2015. Visualizing genome and systems biology: technologies, tools, implementation techniques and trends, past, present and future. *GigaScience* 4:38. DOI: 10.1186/s13742-015-0077-2.

Salomonis N., Hanspers K., Zambon AC., Vranizan K., Lawlor SC., Dahlquist KD., Doniger SW., Stuart J., Conklin BR., Pico AR. 2007. GenMAPP 2: new features and resources for pathway analysis. *BMC bioinformatics* 8:217. DOI: 10.1186/1471-2105-8-217.

Schade B., Jansen G., Whiteway M., Entian KD., Thomas DY. 2004. Cold adaptation in budding yeast. *Molecular Biology of the Cell* 15:5492–5502. DOI: 10.1091/mbc.E04-03-0167.

Schultheiss SJ., Münch M-C., Andreeva GD., Rätsch G. 2011. Persistence and availability of Web services in computational biology. *PloS One* 6:e24914. DOI: 10.1371/journal.pone.0024914.

Schultheiss SJ. 2011. Ten simple rules for providing a scientific Web resource. *PLoS computational biology* 7:e1001126. DOI: 10.1371/journal.pcbi.1001126.

Shannon P., Markiel A., Ozier O., Baliga NS., Wang JT., Ramage D., Amin N., Schwikowski B., Ideker T. 2003. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research* 13:2498–2504. DOI: 10.1101/gr.1239303.

Shneiderman B., Plaisant C., Cohen M., Jacobs SM., Elmqvist N., Diakopoulos N. 2016. *Designing the user interface: strategies for effective human-computer interaction*. Hoboken: Pearson. ISBN: 978-0-13-438038-4

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.

Tufte, E. 1983. *The Visual Display of Quantitative Information*. Cheshire, Connecticut: Graphics Press. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number) [0-9613921-4-2](https://en.wikipedia.org/wiki/Special:BookSources/0-9613921-4-2).

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.