New data analysis and development of new GRNsight functionality will aid researchers in discovering more about gene regulatory networks

Blair V. Hamilton, Emma C. Tyrnauer, Katherine L. Wright, and Zachary S. Van Ysseldyke

BIOL/CMSI 367-01: Biological Databases

15 December 2017

Introduction

- Generally About Yeast, microarray,
- About GRNSight
- Why this research is important

Methods and Results

It is the integration team's responsibility to uphold the integrity for other's code. Furthermore, through integration, all of the code would be represented on one consolidated webpage. Specifically, the integration team was tasked with coding a right click function and populating the page for the specific gene on the website. Before beginning, it was known that not all of the code would be in pristine condition the first few merges. With that in mind, it was still possible to construct a skeleton for the other team's code so it could easily be inserted into the skeletal framework.

Handling code from backend to front end demands consistency between IDs forcing careful code review. Then, the team is able to work on the right click function. The right click function bridged the existing GRNSight website with the new class designed webpage. Through communication with the page design team, the integration team effectively assimilated the code to represent where they wanted the data to go. Furthermore, the API team instantiated variables and functions which the integration team had to understand before manipulating the API calls.

Data Analyst

Data analysis for wild type microarray data began in week 8. Dr. Dahlquist provided an excel spreadsheet containing the log2 fold changes in gene expression for five time points: 15, 30, 60, 90, and 120 minutes. The yeast were exposed to cold shock at 13°C followed by a 60 minute recovery period at 30°C. An ANOVA test was performed in excel to determine if any genes had an expression change that was significantly different than 0 at any time point after cold shock (specifics of procedure for this analysis can be accessed on *Emmatyrnauer Week 8* individual assignment page) . As p-value values parameters for significance became more

stringent, fewer genes were considered to have significantly changed gene expression following cold shock (Table 1). However, the Benjamini and Hochberg-corrected $p<0.05$ parameter revealed the second largest percentage of genes demonstrating change in expression following cold shock (Table 1).

Table 1. Percentage of genes demonstrating significant changes in gene regulation decreased with stricter ANOVA p-values.

ANOVA	***Wild type***
$p<0.05$	2,528 (40.85%)
$p<0.01$	1,652 (26.70%)
$p<0.001$	919 (14.85%)
$p<0.0001$	496 (8.01%)
Benjamini and Hochberg-corrected $p<0.05$	1,822 (29.44%)
Bonferroni-corrected $p<0.05$	248 (4.01%)

In week 10, STEM (Short Time-series Expression Miner) was utilized to cluster *wild type* genes into groups with similar responses to cold shock (detailed procedure can be accessed on *Emmatyrnauer Week 10* individual assignment page) (Fig. 1). These clusters were ordered based on significance (Fig. 1). While some clusters demonstrated clear downregulation of genes following cold shock (profiles 9 and 0), others demonstrated clear upregulation (profiles 45 and 48) (Fig. 1).

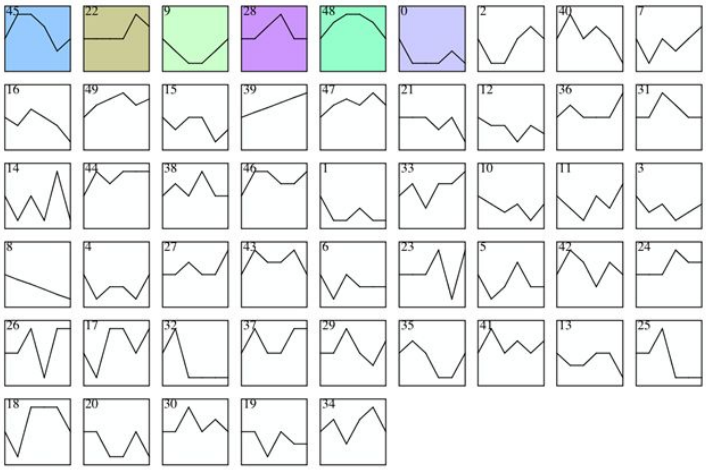


Figure 1. Gene clusters for *wild type* data demonstrating similar changes in gene expression following cold shock. Colored clusters are the most significant. Clustering was performed with STEM software.

Profile 45 cluster was selected for further analysis for various reasons; a large number of genes were assigned to this profile (549) and they demonstrated very clear upregulation following cold shock (Fig. 2). Specifically, genes were upregulated between 0m and 15m, maintained at a constant regulation between 15m and 60m, and then recovered to initial regulation starting at 60m (Fig. 2). Interestingly, some of the genes in this cluster were significantly downregulated following exposure to 30°C (Fig. 2). This suggests that the proteins that are used to maintain viability of the cell following cold shock are unnecessary upon exposure of the cells to warmer temperatures.

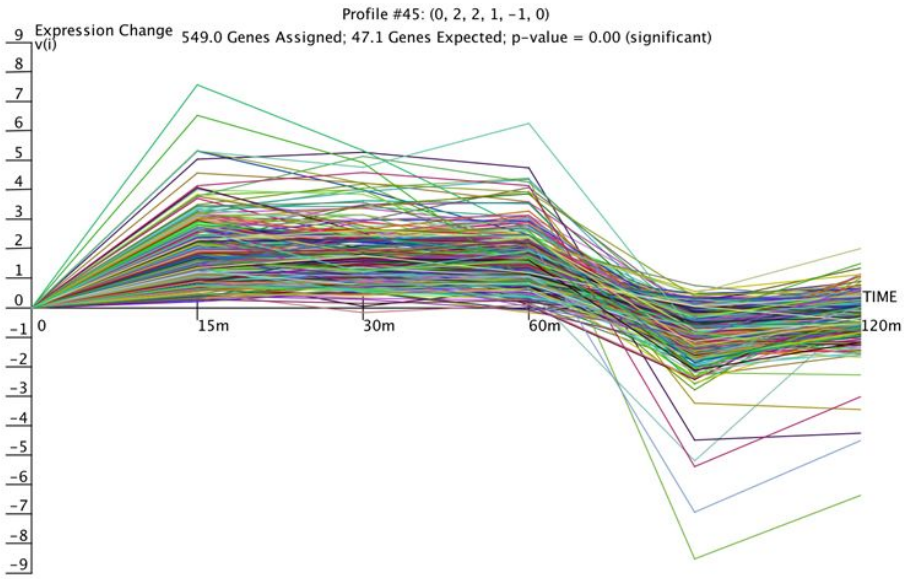


Figure 2. Expression changes for profile 45 genes over time. Yeast cells were exposed to 13°C at 0m and allowed to recover at 30°C starting 60m.

The function of the genes from profile 45 was examined further through analysis of the gene ontology terms associated with this cluster. Only three terms were selected for Table 2 because the list is very extensive. However, the complete file can be accessed from both *Emmatyrnauer Week 10* individual page as well as *Lights, Camera, InterACTION! Deliverables* page. All three terms in Table 2 have similar numbers of genes assigned to them. However, only "nuclear lumen" and "RNA binding" have associated p-values that are considered significant (Table 2). This suggests that while some genes that were upregulated in this profile were involved in cellular component organization, this change in expression was not significantly different compared to that prior to cold shock. On the other hand, "nuclear lumen" and "RNA binding" have very small associated p-values as well as corrected p-values. This is most likely due to the fact that these activities are extremely necessary for survival of the cell in extreme environmental conditions such as cold shock. This does not line up exactly with what Murata et

al. (2006) suggest in *Genome-wide expression analysis of yeast response during exposure to 4 C*. These authors infer that many of the many heat shock proteins are utilized under cold shock due to the fact that they are both exposures to extreme environmental conditions (Murata et al. 2006).

Table 2. Three gene ontology terms associated with Profile 45.

Category name	**# Genes category**	**# Genes assigned**	**# Genes expected**	**# Genes enriched**	**P-value**	**Corrected p-value**	**Fold**
Cellular component organization (GO:0016043)	540	161	162.7	-1.7	0.6	1	1
Nuclear lumen (GO: 0031981)	327	183	98.5	84.5	2.40E-27	<0.001	1.9
RNA binding	215	116	64.8	51.2	6.30E-15	<0.001	1.8

The next steps of data analysis required examining which transcription factors were utilized in this cluster of genes that resulted in similar regulation. The Yeastract database was able to identify the most significant transcription factors (with the smallest p-values) involved in the expression of these genes. The initial network that was produced was too large and needed to be trimmed. This was completed by removing less significant transcription factors one by one and visualizing the network in GRNsight (to ensure that there were no free floating genes). The final group of transcription factors which were visualized in GRNsight and their associated p-values are represented in Table 3.

Transcription factor	**NDT80**	**YAP1**	**PDR3**	**UME6**	**MIG2**	**ZAP1**	**HAP4**	**CIN5**
P-value	9.74E-10	1.31E-09	2.82E-09	6.12E-09	1.96E08	2.99E-03	3.02E-02	0.806
Transcription factor	**YOX1**	**SFP1**	**YHP1**	**SUT1**	**STB5**	**ACE1**	**MSN2**	**GLN3**
P-value	0	0	0	7E-15	7E-15	1.9E-14	2.72E-13	9.15E-1

								1

Table 3. Most significant transcription factors for profile 45 with associated p-values.

Transcription factors from Table 3 were utilized to generate a regulatory network with Yeastract; however, this network file was in table format. To better visualize the network, it was opened in GRNsight, which produced Figure 3. While this map allows for easier visualization of the network compared to Yeastract's table, it fails to detail the nature and magnitude of the interaction between the genes (i.e. whether or not the interaction is strong or weak, and induction or repression). As a result, Matalab was utilized to generate a weighted network output file, which when opened in GRNsight, produced Figure 4.

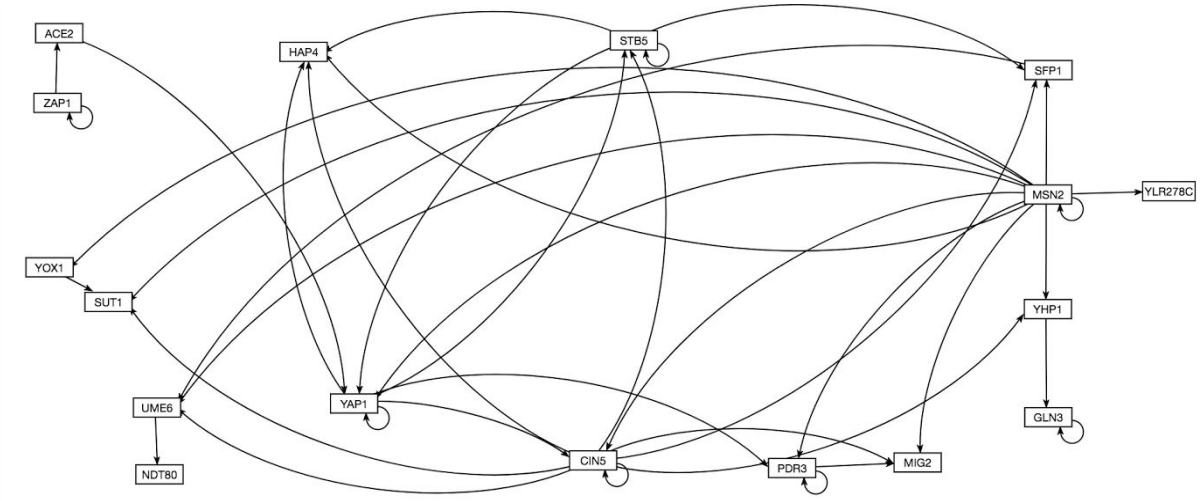


Figure 3. Unweighted gene regulatory network generated with GRNsight

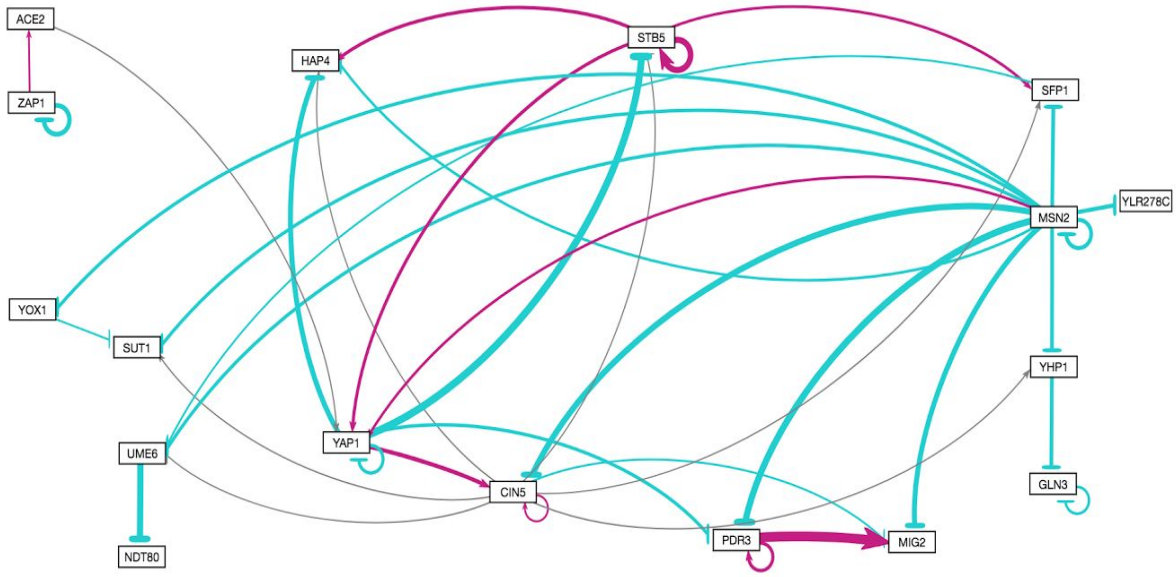


Figure 4. Weighted version gene regulatory network from Figure 3 generated with GRNsight. Weighs were determined using Matlab. Blue lines with terminating bars indicate repression of the gene at the end of the bar, while pink arrows indicate induction of genes.

The gene regulatory network from Figure 4 reveals that most interactions involved repression of other genes. MSN2 represses 10 genes, including itself, indicating its importance in this regulatory network (Figure 4). On the other hand, STB5 is only involved in induction of 4 genes, including itself (Figure 4). While some genes are only repressed by genes belonging to the network, others are both induced and repressed; for example, CIN5 is induced by YAP1 but repressed by MSN2 (Figure 4). This may suggest a system of "checks and balances" so that genes are not overly expressed or underexpressed.

QA

Table 1--JASPAR	
Data	Reasoning
Class	This information is relevant to all genes, so it probably shouldn't have been retrieved from JASPAR. It is suggested that the gene page is re-coded to retrieve this data from a different database.
Family	
Matrix ID	JASPAR only contains information about genes that code proteins that act as transcription factors. This information is only relevant for transcription factors, so it is appropriate to use JASPAR for these elements. Matrix ID is unique to JASPAR, helps user find the gene on JASPAR.
Sequence Logo	
Frequency Matrix	

Table 2--NCBI	
Data	Reasoning
NCBI Gene ID	So users may easily find this gene in NCBI database.
Locus Tag	This information can be easily found in the summary section of a gene page within NCBI, which the project managers concluded would be easy for the coders to retrieve.
Also Known As	
Chromosome Sequence	NCBI has an extensive library of reference sequences. The project managers believed that this database would be most reliable in this regard.
Genomic Sequence	

Table 3--Ensembl	
Data	Reasoning
Gene ID	So users may easily find this gene in Ensembl database.
Description/function	
DNA sequence	
Gene Location	

Gene Map	

Table 4--Uniprot	
Data	Reasoning
Uniprot Gene ID	So users may easily find this gene in Uniprot database.
Protein Sequence	Uniprot is known for its information on proteins, so the Project Managers decided that this database was the most appropriate choice
Similar Protein	
Protein type/name	
Species	

Table 5--SGD	
Data	Reasoning
SGD Gene ID	Standard Name, Systematic name, SGD ID
Regulation -Regulators -targets	
Interaction -Total interactions -Physical interactions -Genetic interactions	
Gene ontology	Summary, Molecular function (x2) Biological process, Cellular component (x2)

Coders

In order to make the GRNsight website more useful, Zach Van Ysseldyk and Blair Hamilton were placed in charge of the coding portion for the Interaction and Integration Team. Their work began Week 11 of the course with a journal club presentation on “The secret startup that saved the worst website in America” article written by Robinson Meyer. The article discusses the work of a small team in charge of fixing the Healthcare.gov website. Healthcare.gov had many technical issues such as frequent crashes, and it was difficult for users to navigate.

During the journal club presentation held during Week 12, Zach and Blair discussed the relationship between this article and the hopes for the GRNsight project. One of the many takeaways from the article is the importance of communication with all stakeholders, in particular coders, data analysts, and project managers. The article mentions different communication pathways used, such as email and “HipChat,” an instant messaging application for the work environment. This proved powerful: when we reached roadblocks, using multiple communication platforms allowed for easy access for questions and coding assistance both between Zach and Blair,and with other group members, coders and Drs. Dahlquist and Dionisio.

Following the Week 12 presentation, Zach and Blair created a fork off of the GRNsight github open source code and then collaborated with all other coders on their respective teams. Once collaboration forks were created Zach and Blair began work on creating a right-click functionality for the GRNsight page used for when a user clicks on a gene.

The right click function bridges the existing GRNSight website to the newly developed gene page. First, the user lands on a gene regulatory diagram. If the user desires more information on the gene, they may right click on the gene node to open up a new tab with the

information about the gene. First, before even thinking about the gene page, one must code a function so that the right click opens up a new tab. Below one may find the code for the whole right click function.

```
.on("dblclick", nodeTextDblclick)
.on("contextmenu", function (gene) {
    console.log(gene);
    var tempLink = $("<a></a>")
        .attr({
            href: "/gene/info.html?" + $.param({symbol: gene.name}),
            target: "_blank"
        });
    $("body").append(tempLink);
    tempLink.get(0).click();
    tempLink.remove();
    d3.event.preventDefault();
});
```

The `"target: "_blank";"` line is supposed to open up a new tab. However because of security reasons, the browsers denied this request to open up a new tab. With the help of Professor Dionisio, the integration team overcame this problem by creating a fake anchor. The fake anchor, denoted as: `var tempLink = $("<a></a>")` acts as a temporary landing page for our tab. Then, the page is able to land. After the page lands, the anchor is taken away denoted as: `tempLink.remove();`. The actual link that the user will see is noted as the "href." A href consists of a database template URL which is then concatenated by tagging on the specific gene that the user clicks on.

After the right-click function was working appropriately, the next milestone for the coders was to identify and integrate the NCBI, Ensembl, SGD, UniProt and JASPAR database data. Through the work of the gene database API and JASPAR teams, API functions were implemented to retrieve the desired data requested by the project managers and biologists in the class. This data needed a desirable layout and functionality to it which was implemented by the

page design team to create an easy-to-use website for a central database data location. But these teams needed to come together in order to have a productive page, so this is where the interaction and integration team comes in. Zach and Blair were given two types of code, HTML and Javascript (js), in order to weld together the three groups' work. This was done through an info.js file that stored variable links between the data classes on the HTML page and the API calls set up by the API and JASPAR teams. Below is an example of the code that links these files:

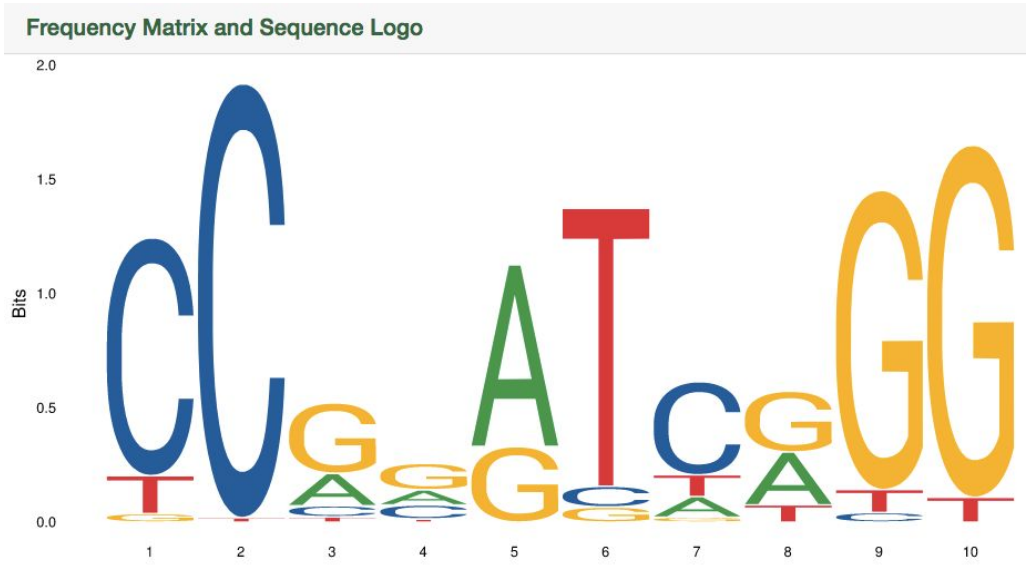
```
var api = window.api;
api.getGeneInformation(obj.symbol).done(function (gene) {

    var sgdHrefTemplate = "https://www.yeastgenome.org/locus/";
    var sgdId = gene.sgd.sgdID;
    $(".sgd-link").text(sgdId).attr({ href: sgdHrefTemplate + sgdId });
```

The `var api` represents the overall api object that will contain all the necessary data points and links. The `getGeneInformation(obj.symbol)` represents the function the API teams made in order to retrieve the data from each desired database. Similar to the picture above, each database was given an HrefTemplate that contained the database's generic gene page URL. Once this template was set up almost every variable link utilized the template to reference that database's URL. For the picture above an `sgdID` was created to retrieve sgd specific numbering system, their "ID", for the gene that was selected and right-clicked. `Gene.sgd.sgdID` represents the pathway taken to retrieve the id from the overall gene object, then the sgd api calls, then finally the sgdID location named by the API team. Lastly, the third code portion: `$(".sgd-link").text(sgdId).attr({ href: sgdHrefTemplate + sgdId });`, represents the HTML class name set up by the page design team, next the data retrieved and finally the URL needed to reference the database plus the gene id. The picture below represents

the website in its current form displaying the variable links, ids being shown and tabs with the desired data within it.

Every variable link has the above layout except for the sequence logo and the frequency matrix. The sequence logo required an img class on the info.html page in order to implement the image to appear on the website. The code to display this information and how it appears on the site is below:



(sequence logo)

```
// Fequency Matrix and Sequence Logo
var sequenceLogo = gene.jaspar.sequenceLogo;
$(".sequenceLogo").attr({ src : sequenceLogo });
```
(info.js)

```
<div id="jasparInfo" class="collapse" role="tabpanel" aria-labelledby="headingNine">
  <div class="card-block">
    <img class="sequenceLogo" src="">
    <div class="container">
```
(info.html)

As described above the info.html file contains an img class that is different that the p classes for the gene ids and other variable links. The other difference revolves around the attr. Tag having a src call instead of an href. When talking with the JASPAR coding teams it was explained that their URL was already attached to their API call for the sequenceLogo so the only request needed was the name of the call.

The second variable link that is different is the frequency matrix which is in the form of a table. The code looks as follows:

```
var frequencyMatrix = gene.jaspar.frequencyMatrix;
var a = "";
for (var i = 0; i < frequencyMatrix.A.length; i++) {
    a += "<td>" + frequencyMatrix.A[i] + "</td>";
}
$(".frequencyOfA").append($(a));
```
(info.js)

```
<table class="frequencyMatrix table table-dark table-striped">
  <thead>
    <tr class="frequencyOfA">
      <th>A:</th>
    </tr>
  </thead>
  <tbody>
    <tr class="frequencyOfC">
      <th>C:</th>
    </tr>
    <tr class="frequencyOfG">
      <th>G:</th>
    </tr>
    <tr class="frequencyOfT">
      <th>T:</th>
    </tr>
  </tbody>
</table>
```
(info.html)

In order to preserve the table look of the JASPAR site, the page design team and the interaction and integration team collaborated on this table layout. For the injo.js file the coders created separate for loops in order to parse through the frequency matrix data as it could range between 0-15 integer results. Once the for loops were created the coders created the link by appending to each A,C,G, and T class. The finished product looks as follows, this particular example is for the gene ASH1:

A:	0	0	61	58	130	0	33	80	0	0
C:	220	291	21	54	0	11	150	0	8	0
G:	8	0	133	101	54	8	8	92	249	256
T:	35	3	10	12	0	147	35	26	19	18

Overall, the GRNsight page now possesses a right-click function that brings the user a new tab that contains gene information. The gene page contains data from the following sources: NCBI, SGD, Ensembl, UniProt and JASPAR. This data has been organized and carefully chosen by

each Biological Database team, and all necessary code files and information has been pushed to the GRNsight github page.

Conclusions

Acknowledgements

We would first like to thank our professors, Dr. Kam D. Dahlquist and Dr. John David N. Dionisio, for their help and support throughout this project; we learned much about databases, coding, research best practices, and much more.

We would also like to thank all of our fellow students in the BIOL/CMSI 367-01: Biological Databases course. We worked with each member of the class at some point during this semester, and we learned from each other in ways which impacted our final project. We extend particular thanks to the following students: Eddie Azinge and Eddie Bachoura, for their coding expertise; Hayden Hinsch, Quinn Lanners, and Corinne Wong, for their support of our own Project Manager;

References

Gene Ontology Consortium. (2017). The Gene Ontology. Retrieved November 19, 2017, from http://geneontology.org

Get url parameter jquery Or How to Get Query String Values In js. Retrieved December 04, 2017, From https://stackoverflow.com/questions/19491336/get-url-parameter-jquery-or-how-to-get-query-string-values-in-js

GRNsight (2017) Retrieved December 4, 2017, from http://dondi.github.io/GRNsight/

How to retrieve GET parameters from javascript? Retrieved December 04, 2017, from https://stackoverflow.com/questions/5448545/how-to-retrieve-get-parameters-from-

javascript

JavaScript Window Location. Retrieved December 04, 2017, from https://www.w3schools.com/js/js_window_location.asp

LMU BioDB 2017. (2017). Coder. Retrieved November 28, 2017, from https://xmlpipedb.cs.lmu.edu/biodb/fall2017/index.php/Coder

LMU BioDB 2017. (2017). Week 8. Retrieved October 23, 2017, from https://xmlpipedb.cs.lmu.edu/biodb/fall2017/index.php/Week_8

LMU BioDB 2017. (2017). Week 9. Retrieved November 19, 2017, from https://xmlpipedb.cs.lmu.edu/biodb/fall2017/index.php/Week_9

LMU BioDB 2017. (2017). Week 10. Retrieved December 4, 2017, from https://xmlpipedb.cs.lmu.edu/biodb/fall2017/index.php/Week_10

Matlab. Retrieved December 7, 2017, from LMU's computer lab https://www.mathworks.com/products/matlab.html

Meyer, R. (2015, July 09). The Secret Startup That Saved the Worst Website in America. Retrieved November 7, 2017, from https://www.theatlantic.com/technology/archive/2015/07/the-secret-startup-saved-healthcare-gov-the-worst-website-in-america/397784/

Murata, Y., Homma, T., Kitagawa, E., Momose, Y., Sato, M. S., Odani, M., ... & Fujita, K. (2006). Genome-wide expression analysis of yeast response during exposure to 4 C. *Extremophiles*, 10(2), 117-128.

Short Time-series Expression Miner (STEM). (2006). Retrieved November 19, 2017, from http://www.cs.cmu.edu/~jernst/stem/

Window.location. Retrieved December 04, 2017, from https://developer.mozilla.org/en-US/docs/Web/API/Window/location

YEASTRACT. Retrieved December 4, 2017, from http://www.yeastract.com/formgenerateregulationmatrix.php