Generating Information of Genes from Databases Relating to Cold-Shock Expression Responses

Eddie Azinge, Dina Bashoura, John Lopez, Corinne Wong

BIOL 367 Biological Databases

Loyola Marymount University, Los Angeles, CA, USA

December 15, 2017

Introduction.

Genes are able to change their expression, turning on or off, and they are able to influence the expression of other genes. There is still a need to understand the interactions and influences between genes of organisms. Moreover, there is more to learn about how the genome responds to environmental conditions to aid organisms in survival.

Saccharomyces cerevisiae, yeast, is known to respond to cold shock. Yeast gene expression up or down regulates in response to cold shock, meaning they increase or decrease their expression. Several clusters of genes form that cooperatively regulate, relating to specific responses of the organism. For example, expression of genes that are essential factors for transcription and processing were up-regulated in response to cold shock. In another example, the expression of genes related to trehalose and glycogen production were up-regulated to increase the organisms energy preserves. Overall, there was a clear trend of expression changes. The genes first changed expression to help transcription and translation, so the organism can make proteins to help maintain the integrity and basic functions of cells. Next, the expressions of genes changed to improve tolerance to the new environmental condition. Stress response induced genes showed up-regulation of expression in yeast cells in response to cold shock, which allowed them to adapt and gain tolerance to the low temperature (Sahara, et al., 2002).

Yeast cold shock microarray data collected from Dr. Dahlquist's lab, testing the responses of yeast cell genes to cold shock, will be analyzed. The data will be searched for significant changes in expression (up or down regulation) of genes and in which clusters or areas of function in the cell. This analysis will give further information about the influences certain

yeast cell genes have and how they react to cold shock. This newly analyzed data can be compared with that of Sahara, et al. (2002) to gain more confidence in the current understanding and knowledge of gene interactions.

There is previous research on the subject, like cold shock response of yeast genes, but there is still information that is unknown. It is still unknown which transcription factors regulate and control responses to cold shock in yeast. It is also still unknown why some genes are up or down regulated. For example, in response to cold shock, yeast gene expressions of HSP12 and HSP26 were up-regulated. This is peculiar because HSP12 and HSP26 are heat shock protein genes, and yet they were up-regulated in response to cold shock.

To learn more about these unknown aspects of gene responses, there are useful tools that can help reveal more information and answers, like GRNsight. GRNsight allows researchers to see the connections between genes. Researchers can see which genes positively influence each other and which genes negatively influence each other. This tool will help researchers learn more about gene interactions and allow them to connect gene interactions to responses to stresses.

GRNsight is already a useful tool, but what further information would be valuable to researchers when looking at a gene regulatory network? This project focused on which information would be helpful to provide to GRNsight users. Important information was selected from several databases, including UniProt, Saccharomyces Genome Database (SGD), Ensembl, NCBI, and JASPAR. Code was created to pull this information from the various databases and present it on the website for users to easily access. This information allows the users to learn more about the certain genes right from the program, which is extremely useful and allows for more efficiency when learning about the interactions and roles of genes.

Materials & Methods.

Data Analyst.

The job of the data analyst began by working with the project manager/quality assurance to research articles on *S. Cerevisiae,* commonly named yeast. Sites like the National Center for Biotechnology Information (NCBI), PubMed, Google Scholar, and Web of Science were used as sources to research information on yeast and their effects on cold shock, ensuring that every article found contained three topics: yeast, cold shock, and microarray data. Three articles per team (Data Analyst and Quality Assurance) were generated, totalling in 6 articles that could be used as references for our future analyses throughout this project. After consulting Dr. Dahlquist about the 6 articles, one of the six was chosen to focus on and present in a journal club the following week. This article was written by Sahara, Goda, & Ohgiya and titled "Comprehensive expression analysis of time-dependent genetic responses in yeast cells to low temperature" (2002). For a deeper explanation of methods, refer to Dina Bashoura's Week 11 electronic notebook (LMU BioDB, 2017).

After presenting in journal club, the analysis portion of this project began. Data from Dr. Dahlquist's lab on yeast and cold shock was provided in excel files with multiple deletion strains of yeast genes that are thought to be involved in gene expression. The gene dGLN3 was assigned to our team for further analysis. Each gene on the spreadsheet contained microarray data for expression rates at time points 15, 30, 60, 90, and 120. Initially, the data for dGLN3 was processed by performing a within-strain Analysis of Variance (ANOVA) to determine if any genes had a gene expression change that was significantly different than zero at any timepoint.

Once this ANOVA was done on excel, the data was filtered to only show significant results with a p value of < 0.05. This information told us which genes were being upregulated and which were being down regulated. The significance was determined by performing adjustments to the p value by correcting for the multiple testing problem.This was done by calculating the Bonferroni p value as well as the Benjamini & Hochberg p value. The Bonferroni adjustment compensates for the increasing likelihood of incorrectly rejecting a null hypothesis caused by testing multiple hypotheses. The Benjamini & Hochberg correction produces more stringent results than the Bonferroni p value, allowing for a more confident p value. The results were then compared to the wild type strain used in this research and presented in Table 1. For a deeper explanation of methods, refer to Dina Bashoura's Week 8 electronic notebook (LMU BioDB, 2017).

Once an ANOVA was performed, the microarray data was prepared to be put into Short Time-Series Expression Miner (STEM). This Java program allows for the clustering, comparing, and visualizing of short time series gene expression data from microarray experiments (Earnest & Bar-Joseph, 2006). STEM provides the opportunity for researchers to identify significant temporal expression patterns profiles and the genes associated with these profiles. To prepare the data, the insignificant Benjamini & Hochberg p values were deleted, leaving only the significant data timepoints. The average log fold change for each timepoint was then imputed into STEM and run. STEM generated a Gene Ontology (GO) list for sets of genes having the same temporal expression pattern to easily visualize the behavior of genes belonging to a given GO category. One of the 7 significant profiles generated were used for further analysis. Figure 1 shows the significant profiles generated from STEM. The implication is that the genes in this profile share the same expression pattern because they are regulated by the same (or the same set) of

transcription factors. Table 2 in results shows 3 of the 406 GO terms assigned to that profile. For a deeper explanation of methods, refer to Dina Bashoura's Week 10 electronic notebook (LMU BioDB, 2017).

The final step in analyzing the microarray data was to utilize Yeastract to infer which transcription factors regulate the profile chosen from STEM. Yeastract is a curated repository of 163,000 regulatory transcription factors in yeast (Teixeira, 2013). The GO list generated from STEM was imputed into Yeastract to determine a list of transcription factor candidates that are most likely associated with the gene functions in the profile. The output file gave this list of transcription factors ranked by significance. For a deeper explanation of methods, refer to Dina Bashoura's Week 14 electronic notebook (LMU BioDB, 2017).

The next step was to utilize GRNsight to visualize the transcription factors in a network. This process began by choosing 15 significant transcription factors from the output file from Yeastract, which can be seen with their p values on Table 3. This list was used to generate a regulatory matrix with the help of Yeastract once again. Once this list was imputed into Yeastract, it generated a mathematical output file which was later formatted to be used in GRNsight to visualize the matrix (Dahlquist et. al., 2016). This allowed for the visualization of the matrix with unweighted transcription factors chosen, meaning the magnitude and direction of the transcription factors in relation to each other were not incorporated in this matrix (Figure 3). In order to view the magnitude and direction, MATLAB was utilized to run GRNmap which allows for the visualization of these properties. The output file from GRNmap was put back into GRNsight and resulted in a colored matrix showing the upregulatory and down regulatory qualities of the transcription factors, as well as the magnitude and direction of them (Figure 4).

For a deeper explanation of methods, refer to Dina Bashoura's Week 14 electronic notebook (LMU BioDB, 2017).

Quality Assurance.

The quality assurance guild worked together to decide which information was useful and important. An initial list was made of information related to genes that seemed important. Then, the five databases (UniProt, SGD, Ensembl, NCBI, and JASPAR) were examined and scanned for the most valuable information. Another list was made, dividing the desired content into separate lists for each database. This list was then given to the coders, so they knew what to pull.

Coders.

The process of coding the API functionality can be characterized by setting up, researching, collaborating, and implementing. Our job was to essentially convert data given to us in document form and extract pieces of the data that was requested by the other teams. As coders, we needed to ensure that our personal computers were set up to handle this task and our version control environment was proficient, and the first 3 milestones ensured this. Milestone 1 allowed us to make sure we had Node.js, Atom, Git, Google Chrome Developer Tools, and CURL installed for working. Milestone 2 allowed us to create a copy of the GRNSight project for us to work on without modifying the public copy. Finally, Milestone 3 allowed us to ensure that we were using version control through Git properly and that it would allow us to edit GRNsight as necessary.

Once our environments were set up, we had to obtain an understanding of the task ahead of us through research and collaboration. The Week 9 assignment provided us with a guideline on how to implement the process for extracting the XML/JSON data from the particular website.

We decided to have the code return an object while the other teams could take parts of this object for them to use. This object would consist of data pulled from NCBI, UniProt, SGD, Ensembl, and JASPAR. The JASPAR team needed an understanding of how this object would work in order for them to integrate their code into ours.

```
    };
    return {
        jaspar: parseJaspar(jasparInfo),
        ncbi: parseNCBI(ncbiInfo),
        ensembl: parseEnsembl(ensemblInfo),
        uniprot: parseUniprot(uniprotInfo),
        sgd: parseYeastmine(yeastmineInfo),
    };
};
```

From this baseline, we then focused on interacting with the API's that were provided from the Week 9 assignment. At the beginning, we implemented these API calls outside of the context of the GRNsight project, using ES6 language constructs in order to represent the functions more succinctly. This allowed us to replicate the results of the Week 9 bash commands by following a sequential paradigm, without fully sacrificing the control of our code, through the use of javascript's async and await. We quickly realized, however, that GRNsight was operating on ES5; a version of javascript that is devoid of many of the newer language changes, in particular: the introduction of async and await. With this shift, we needed to backport our code to instead work on an ES5 project, which was an unexpected development. Thankfully, however jQuery provides a nice interface for asynchronous programming in its deferred objects. Through these, we were able to adapt our code into a form that allowed us to more easily make the changes that we needed in order to adapt to the constraints of the GRNsight project. An example of one of the functions that we ended up with can be seen below.

```
var getUniProtInfo = function (geneSymbol) {
    return $.get({
        url: "http://www.uniprot.org/uploadlists/",
        data: {
            from: "GENENAME",
            to: "ACC",
            format: "tab",
            taxon: "559292",
            query: geneSymbol,
        },
        dataType: "text",
        timeout: 5000,
    }).then(function (data) {
        var regex = new RegExp(geneSymbol + "[ \t\r\n\v\f]*([A-Z0-9]+)", "gm");
        var id = regex.exec(data)[1];
        return $.get({
            url: "http://www.uniprot.org/uniprot/" + id + ".xml",
            timeout: 5000,
        });
    });
};
```

While we setup the data extraction, Quality Assurance provided us with a list of the data we needed to find within the XML/JSON and extract. Once we had the data coming from where we wanted it to, it was now a matter pulling these selected pieces of data. Extracting the XML was a challenge because while some of the data could easily be accessed by calling the name of an XML tag, clever manipulation of the data was necessary to retrieve everything. Some data could not be called by tag for example, so we had to manipulate the Document Object Model with the understanding that you could call nodes by their relation to another node. This was seen in Uniprot below, when to get the proteinType for the web page, we had call the necessary node by its relation to another node.

```
// change if any preprocessing needs to be done on the data before being given to the application
var filterData = function (uniprotInfo, ncbiInfo, yeastmineInfo, ensemblInfo, jasparInfo) {
    var parseUniprot = function (data) {
        return {
            uniprotID: XMLParser(data.getElementsByTagName("name")[0]),
            proteinSequence: XMLParser(data.getElementsByTagName("sequence")[0]),
            proteinType: XMLParser(data.getElementsByTagName("protein")[0].childNodes[1].childNodes[1]),
            species: XMLParser(data.getElementsByTagName("organism")[0].childNodes[1]),
        };
    };
```

In other cases, the data was not directly given in one XML tag, but several. The genomic sequence, for example, displayed as “Chromosome: XVI; NC_001147.6 (159548..160594)” on NCBI is given as the following XML code:

```
<Mim></Mim>
▼<GenomicInfo>
 ▼<GenomicInfoType>
    <ChrLoc>XVI</ChrLoc>
    <ChrAccVer>NC_001148.4</ChrAccVer>
    <ChrStart>919380</ChrStart>
    <ChrStop>920486</ChrStop>
    <ExonCount>1</ExonCount>
  </GenomicInfoType>
```

Thus, in order to have the data appear as it does on NCBI, it requires the extraction of several tags and returning of a new string object below.

```
var parseNCBI = function (data) {
    var tagArray = serializer.serializeToString(data.getElementsByTagName("OtherAliases")[0]).split(","
    return {
        ncbiID: data.getElementsByTagName("DocumentSummary")[0].getAttribute("uid"),
        locusTag: tagArray[0].replace(/\<.*?\>\s?/g, ''),
        alsoKnownAs: tagArray.slice(1).join().replace(/\<.*?\>\s?/g, ''),
        chromosomeSequence: XMLParser(data.getElementsByTagName("ChrAccVer")[0]),
        genomicSequence: XMLParser(data.getElementsByTagName("ChrLoc")[0]) + "; "
        + XMLParser(data.getElementsByTagName("ChrAccVer")[0]) + " ("
        + XMLParser(data.getElementsByTagName("ChrStart")[0])
         + ".." + XMLParser(data.getElementsByTagName("ChrStop")[0]) + ")",
    };
};
```

In addition to selecting XML tags, we had to implement the use of regular expressions to eliminate any sort of code that resided within the XML tags in order to get a clean string. Another particular challenge was when the desired data for multiple points could be located in a single XML tag , such as the case when the LOCUS Tag and Other Aliases were found in the following XML tag: <OtherAliases>YPR191W, COR2, UCR2</OtherAliases>. To overcome this challenge, it was necessary to take the data as a string, split the string into an array, and return only parts of the array needed for the desired data. The extraction of JSON data was more

straightforward, as we had to implement values of the JSON objects given to us as elements of our own object. Ultimately, this was the last portion of work that we needed to do in order to bring the project to completion.

```
var parseYeastmine = function (data) {
    return {
        sgdID: data.primaryIdentifier, // string
        standardName: data.symbol, // string
        systematicName: data.secondaryIdentifier, // string
```

.

Results/Discussion.

Data Analyst.

The original excel file provided by Dr. Dahlquist's lab on yeast and cold shock had 4 replicates for each time point, totaling in 20 data points. There was a total of 6189 genes used in this experiment and included in the excel spreadsheet. 6652 cells were replaced because they contained no data. The purpose of the within-stain ANOVA test is to determine if any genes had a gene expression change that was significantly different than zero at *any* time point. The ANOVA p values, Bonferroni p value, and Benjamini & Hochberg p values can be seen in this table below.

Table 1: This table shows the results from the ANOVA run on the deletion strain dGLN3 in comparison to the wild type strain. The decreasing p values increase the significance of the gene expression change. The lower the p value, the more confidence there is with the data.

ANOVA	WT	*dGLN3*
p < 0.05	2528 (40.85%)	2135 (34.50%)
p < 0.01	1652 (26.70%)	1204 (19.45%)
p < 0.001	919 (14.85%)	514 (8.31%)
p < 0.0001	496 (8.01%)	180 (2.91%)
Benjamini & Hochberg-corrected p < 0.05	1822 (29.44%)	1185 (19.15%)
Bonferroni-corrected p < 0.05	248 (4.01%)	45 (0.73%)

These multiple p values allow for the stringency of the data, which increases the significance of the gene expression change. I performed 6189 hypothesis tests, expecting to see a gene expression change for at least one of the timepoints by chance in about 5% of our tests, or 309 times when using a p value of < 0.05. Since I have more than 309 genes that pass this cut off, I know that some genes are significantly changed. However, I don't know *which* ones. The final outcome of this data shows that 45 genes out of 6189 total genes had a p value of < 0.05, meaning that they are significant. Specifically, this means that 0.73% of the data was significant.

After retrieving the ANOVA results, the Bonferroni and Benjamini & Hochberg p values were filtered to only show the significant values. This resulted in 1258 genes after the filtering. After inputting these values into STEM, 7 significant profiles representing clusters of genes were presented. Figure 1 below shows the results.

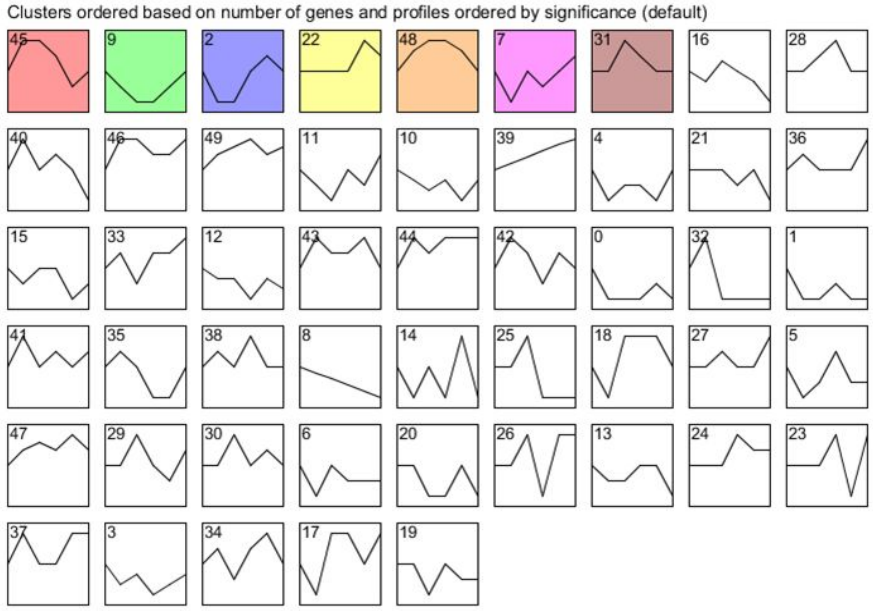

Figure 1: This figure illustrates the 7 significant profiles generated from STEM. Each profile represents a cluster of genes showing the same temporal expression pattern.

For our project, we focused on Profile #45 due to its upregulation and downregulation pattern as well as the number of genes associated with this profile and its significance. Each profile is integrated with a GO list and a gene list that yields its information from other databases. Profile #45 has 406 genes assigned, 29.9 genes expected, and a p value of 0.00 meaning it is significant. Figure 2 illustrates how at the first time point, 15 minutes, all the genes were up regulated in the same pattern, then around time point 60, were all down regulated until they were stabilized. Since time point 60 was when the cold shock stopped and heat was then applied to bring the temperature to normal level, we can conclude that the genes in this cluster are upregulated when exposed to cold shock and are downregulated or leveled off when exposed to hot temperatures.

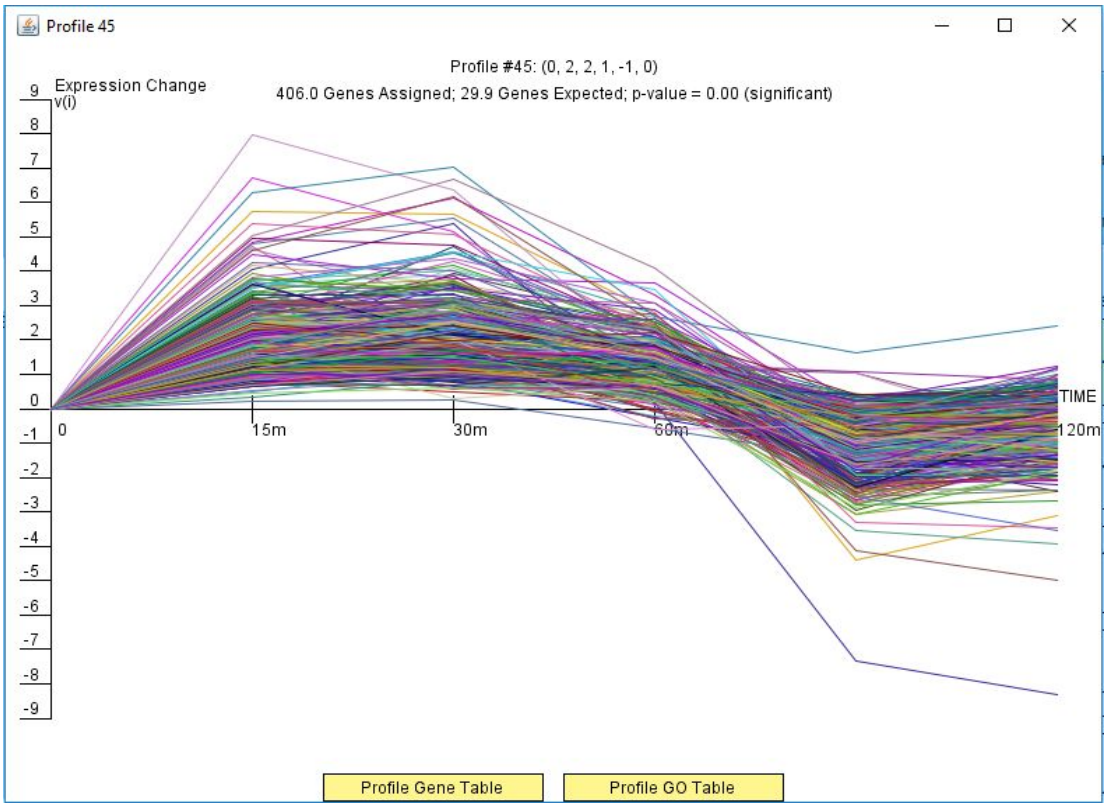

Figure 2: This figure shows the expression pattern of genes in profile 45 that was generated from STEM. The axes are expression change over time. Each line corresponds to a gene and all these genes were clustered based off their similar expression pattern.

A preview of the GO list can be seen in Table 2. Three GO terms represent 3 possible functions that the genes in profile #45 have in common. These terms all relate to the mechanism of gene expression. The first GO term shows that there are 189 genes assigned in this cluster to the function of gene expression, with a corrected p value of < 0.001. This means that the number of genes enriched were significantly more than what was expected. Gene expression is defined as the process in which a gene's sequence is converted into a mature gene product or products (proteins or RNA). This includes the production of an RNA transcript as well as any processing to produce a mature RNA product and the translation of that mRNA into protein (Consortium, 2017). This ultimately affects the amount of protein produced which affects the upregulation or downregulation of a gene. The more gene expression that takes place, the more that gene is upregulated.

RNA processing, the second GO term, is defined as any process involved in the conversion of one or more primary RNA transcripts into one or more mature RNA molecules (Consortium, 2017). This is similar to the GO term gene expression in that with the increased production of RNA processing, more genes relating to that function are being produced, meaning that those genes are being upregulated. There are 115 genes assigned to this function with a corrected p value of < 0.001, meaning that the number of genes enriched were significantly more than what was expected.

RNA phosphodiester bond hydrolysis is defined as the RNA metabolic process in which the phosphodiester bonds between ribonucleotides are cleaved by hydrolysis (Consortium, 2017). Because there is an upregulation of the hydrolysis, the expression of the genes that code for that RNA will be underproduced due to the breakdown of those bonds. This process can be used to

downregulate a certain gene expression. There are 37 genes assigned to this function in the cluster with a corrected p value of < 0.001, meaning that the number of genes enriched were significantly more than what was expected.

When the genes in this cluster were exposed to cold shock, they responded by upregulating gene expression, RNA processing, and RNA phosphodiester bond hydrolysis. This means that when exposed to extremely stressful environments, the genes respond by expressing more genes through the production of more proteins, developing more mature RNA molecules which yields more proteins, and increasing the breakdown of RNA bonds through hydrolysis, possibly because the genes associated with this function must be downregulated for the survival of the cell during cold shock. Previous research has found that there are clusters of transcription factor genes that are upregulated when exposed to cold shock. These genes include functions like mRNA transcription, RNA processing, and protein folding genes, which correlates to our findings here (Sahara et. at., 2001).

Table 2: This table shows the Gene Ontology (GO) list for profile #45 showing the gene functions in this cluster. Only 3 are shown here but the complete list can be seen in the deliverables section of Dina Bashoura's Week 10 Electronic Notebook.

Category Name	#Genes Category	#Genes Assigned	#Genes Expected	#Genes Enriched	p-value	Corrected p-value	Fold
Gene expression	382	189.0	123.3	+65.7	1.8E-17	<0.001	1.5
RNA processing	170	115.0	54.9	+60.1	1.7E-24	<0.001	2.1
RNA phosphodiester bond hydrolysis	50	37.0	16.1	+20.9	8.0E-10	<0.001	2.3

Yeastract was then used to interpret which transcription factors regulate profile #45's cluster. Yeastract uses a database of transcription factors and compares them to the ones that you input, then generates a list showing which ones are significant and which are not. The results gave us a list of 30 significant candidates that could be transcription factors regulating that cluster. From those 30, 15 were chosen based on their percent in user set, their % in Yeastract, and their p values. The results are shown in Table 3.

Table 3: This table shows the 15 transcription factors and their p values chosen from the 30 significant ones that Yeastract gave as candidates for regulating profile #45.

Transcription Factors	ACE2	CIN5	GLN3	HAP4	MSN2	PDR1	PDR3
P - Values	6.7E-14	0.6462266	1.21E-13	0.004451	0	6.245E-11	8.92E-13

Transcription Factors	SFP1	SWI5	UME6	YAP1	YHP1	YLR278C	YOX1	ZAP1
P - Values	0	2.782E-10	3.323E-11	2.84E-13	0	3.666E-10	0	8.553E-06

Finally, these transcription factors were formatted and put into GRNsight to visualize the network of the gene interaction. Figure 3 shows the GRNsight output network in black and white, showing the unweighted version of which transcription factor interacts with the other.

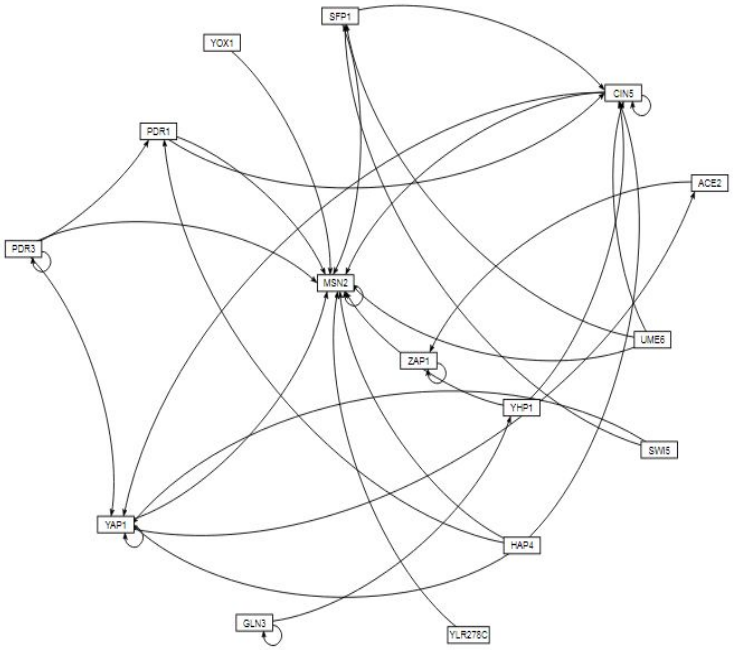

Figure 3: This figure shows the network of transcription factors. Each node represents a gene or transcription factor and each line represents an edge or the way that it interacts with other genes.

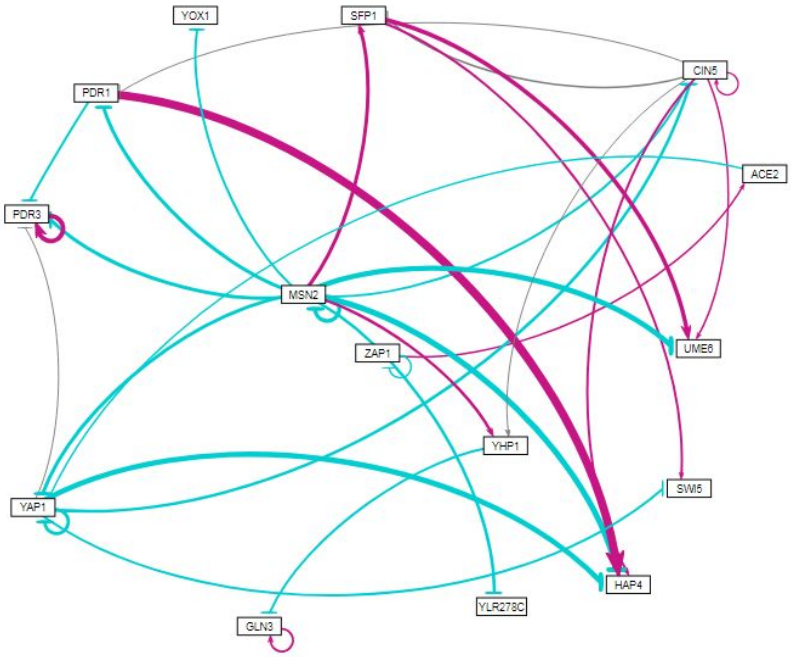

Figure 4: This figure shows the network of transcription factors. Each node represents a gene or transcription factor and each line represents an edge or the way that it interacts with other genes. The thickness of the line represents the magnitude of the interaction. The pink lines represent upregulation and the blue lines represent downregulation.

The network in Figure 4 shows the weighted upregulation and downregulation patterns and interactions of the 15 chosen transcription factors in relation to each other. Visually, you can see that PDR1 to HAP4 has the thickest pink edge, meaning that PDR1 is upregulated and interacts with HAP4 in the network. HAP4 also has thick blue lines connecting to it, meaning that it is also downregulated by other genes like YAP1 and MSN2. There are more blue edges than pink edges, meaning that in this network, there are more transcription factors being downregulated than upregulated.

Quality Assurance.

The information that was chosen from each database is presented below in Table 4. The reasons for why the information was considered important is given in the far right column. The information that was most important or relevant had to do with identifying the gene, characteristics of the gene, and connecting the gene to other gene/cell functions. Of all the information presented on the five databases, this information in Table 4 is believed to be most valuable to researchers who would use GRNsight to learn more about a specific gene and its influences and effects on other genes.

Table 4: Table of content that was chosen from the five databases. The far right column gives the reason why it was chosen and considered important.

Database	**Information**	**Why?**
NCBI	Gene ID	Identifier used by this database; helps user easily find it for further research
	Locus Tag	Another identifier for the gene that helps users easily find it for further research

	Also Known As	Other names it is referred to as, so users will know these other names are referring to the same gene
	Chromosome Sequence	Shows the location on a chromosome and the related genes/close neighbors
	Genomic Sequence	Shows the gene's location in the genome and the related genes/close neighbors
UniProt	Gene ID	Identifier used by this database; helps user easily find it for further research
	Protein Sequence	Shows the proteins that make up the gene
	Protein Type/Name	Tells user which protein it controls/influences
	Species	Tells user with which organism the gene is involved
Ensembl	Gene ID	Identifier used by this database; helps user easily find it for further research
	Description/Function	Tells the user about the genes role(s) and its effects on the cell/organism
	DNA Sequence	Shows which nucleotides code for the particular gene
	Gene Location	Tells users where the gene is located and on which chromosome
	Gene Map	Shows the gene's location in the genome and the related genes/close neighbors
SGD	Gene ID (standard name, systematic name, SGD ID)	Identifier used by this database; helps user easily

		find it for further research
	Regulation (regulators, targets)	Gives the users transcriptional information about the gene, including binding motifs, genes it regulates, and genes that regulate it
	Interaction (total interactions, physical interactions, genetic interactions)	Tells users how the gene interacts with other genes physically or genetically
	Gene Ontology (summary, molecular function, biological process, cellular component)	Provides the GO Annotations, telling user about the gene and its gene product
JASPAR	Matrix ID	Identifier used by this database; helps user easily find it for further research
	Class	Tells users with which aspect the gene is involved
	Family	Gives user further information about the gene's relationship/role in the cell
	Sequence Logo	Gives users a visualization of the frequency of nucleotides in the gene and which nucleotides in an order are more likely to code for the gene
	Frequency Matrix	Gives user data about which nucleotides are most frequent in the gene

Coders.

As discussed above, our final deliverable was a function that can be called through a provided api object's getGeneInformation function. Said function receives a gene symbol as input, and returns a jQuery deferred object which resolves to an object containing the data from the 5 different databases. This object is structured in JSON, and information from a given database can be retrieved through the object's property that matches the database's name.

This process required us to interact with data from a number of perspectives. First, we needed to understand how our classmates each individually retrieved information from the databases they were assigned in Week 9, which required individually going through their assignments and noting the important links, references, and URLs that were used across multiple instances of the same database groups. Once we had a solid understanding of how the data was being made available to us, we then were able to scope out how we would translate this data to instead be made available in javascript, as described above. Dealing with the ES6 vs ES5 debacle required an understanding of the differences between the two versions, and learning to use the data retrieval functionality of jQuery efficiently. Finally, through efficient communication we were able to remain in sync with the Jaspar team, and onboard them onto the framework that we had established for the getGeneInformation function. All in all, this was a cross-disciplinary project that required a lot out of us, but granted us much knowledge in return.

There are a number of properties from the SGD and Ensembl databases that were requested of us, but that we were also unable to retrieve given the data sources that we interacted with from the results of Week 9's homework assignment. Such data sources can be seen below, tagged with the comment: "Information unavailable via regular API".

```
var parseYeastmine = function (data) {
    return {
        sgdID: data.primaryIdentifier,
        standardName: data.symbol,
        systematicName: data.secondaryIdentifier,
        regulators: "N/A", // Information unavailable via regular API
        targets: "N/A", // Information unavailable via regular API
        totalInteractions: "N/A", // Information unavailable via regular API
        affinityCaptureMS: "N/A", // Information unavailable via regular API
        affinityCaptureRNA: "N/A", // Information unavailable via regular API
        affinityCaptureWestern: "N/A", // Information unavailable via regular API
        biochemicalActivity: "N/A", // Information unavailable via regular API
        colocalization: "N/A", // Information unavailable via regular API
        reconstitutedComplex: "N/A", // Information unavailable via regular API
        twoHybrid: "N/A", // Information unavailable via regular API
        dosageRescue: "N/A", // Information unavailable via regular API
        negativeGenetic: "N/A", // Information unavailable via regular API
        phenotypicEnhancement: "N/A", // Information unavailable via regular API
        phenotypicSuppression: "N/A", // Information unavailable via regular API
        syntheticGrowthDefect: "N/A", // Information unavailable via regular API
        syntheticHaploinsufficiency: "N/A", // Information unavailable via regular API
        syntheticLethality: "N/A", // Information unavailable via regular API
        syntheticRescue: "N/A", // Information unavailable via regular API
        geneOntologySummary: data.functionSummary,
        molecularFunction: "N/A", // Information unavailable via regular API
        biologicalProcess: "N/A", // Information unavailable via regular API
        cellularComponent: "N/A", // Information unavailable via regular API
    };
};

var parseEnsembl = function (data) {
    return {
        ensemblID: data.id,
        description: data.description,
        dnaSequence: "N/A", // Information unavailable via regular API
        geneLocation: "N/A", // Information unavailable via regular API
        geneMap: "N/A", // Information unavailable via regular API
    };
};
```

Conclusion.

As discussed in the Week 12 Journal Club Presentation regarding chapter five of Paul Ford's *What is Code?*, the process of developing software is more intricate than simply writing code. Just as Ford explained the complications of developing something perceived to be simple like email validation, this project taught us that something as trivial as displaying text requires collaboration and contains complications throughout the development process. Ford enlightens the reader that although languages will, over time, have implementations in a broader scope, the overall dogma of software development will remain with a significant degree of complexity in development. This was displayed perfectly in our task; despite the use of jQuery making the extraction and parsing of data easier, a significant degree of work went into proper use of these "newer" technologies.

Sahara, et al. (2002), there was a clear trend for the genes to respond to the cold shock by improving their transcription abilities. For example, the genes controlling transcription factors and RNA processing were up regulated. This helps the cell produce more proteins to maintain the integrity and basic functions of the cell. The analysis of the data from Dr. Dahlquist's lab showed similar results. Profile #45 responded in a similar way, up-regulating the genes that had roles in transcription, including RNA processing and RNA phosphodiester bond hydrolysis.

In the future, it would be ideal to make all of the information accessible from one page. This would make it much easier for the user to research the genes of interest. Moreover, once it has been running for a while, research could be done to see what information is actually used by the researchers and/or which information they would prefer to have. The research that has been

done in this project to select the information that is currently displayed is sufficient and will be good to get it started. However, getting the opinions of the actual users allows for the information to be targeted to the actual users of GRNsight. This way, it is much more efficient for researchers to use, so they do not have to locate other information in several different databases. GRNsight can be a single stop for them.

Overall, in this project, initial research was done to get to know more about the subject and learn about gene interactions and yeast gene responses to cold shock. Raw data from Dr. Dahlquist's lab was then taken and analyzed. This provided more understanding of how the data is processed and what it means. Opening the data on GRNsight showed how useful the website can be for visualizing the meaning of the data and connections between the genes. This process of becoming more familiar with the research made the gaps in the current knowledge apparent. Creating a way for GRNsight users to easily access further information about the genes helps them to make the researching process more efficient, and the gene regulatory networks will help them to see the interactions and influences between genes more easily to learn more about them and which genes control genome in responses to stress. Adding this feature to GRNsight makes the researching process much easier and more organized, helping to fill the gaps of what is known about gene interactions and influences.

Acknowledgments.

We would like to thank Dr. Dahlquist for providing the data on yeast and allowing us to manipulate the data from her lab, as well as instructing us and providing help and guidance when needed. We also would like to thank Dr. Dionisio for also providing help and guidance for the

coders as well as correcting assignments and reviewing our work. On top of that, we acknowledge our respective guilds for providing a support system and reference for any questions we had.

References.

Consortium, G. O. (2017). Retrieved December, 2017, from http://amigo.geneontology.org/amigo/search/ontology?q=gene expression

Consortium, G. O. (2017). Retrieved December, 2017, from http://amigo.geneontology.org/amigo/search/ontology?q=rna processing

Consortium, G. O. (2017). Retrieved December, 2017, from http://amigo.geneontology.org/amigo/search/ontology?q=RNA%20phosphodiester%20bond%20hydrolysis

Dahlquist, K. D., Dionisio, J. D. N., Fitzpatrick, B. G., Anguiano, N. A., Varshneya, A., Southwick, B. J., & Samdarshi, M. (2016). GRNsight: a web application and service for visualizing models of small-to medium-scale gene regulatory networks. PeerJ Computer Science, 2, e85.

Ernst, J., Bar-Joseph, Z. (2006). STEM: a tool for the analysis of short time series gene expression data. *BMC Bioinformatics*, 7:191.

LMU BioDB 2017. (2017). Week 8. Retrieved November 29, 2017, from https://xmlpipedb.cs.lmu.edu/biodb/fall2017/index.php/Week_8

LMU BioDB 2017. (2017). Week 10. Retrieved November 29, 2017, from https://xmlpipedb.cs.lmu.edu/biodb/fall2017/index.php/Week_10

LMU BioDB 2017. (2017). Week 11. Retrieved November 29, 2017, from https://xmlpipedb.cs.lmu.edu/biodb/fall2017/index.php/Week_11

LMU BioDB 2017. (2017). Week 11. Retrieved November 29, 2017, from https://xmlpipedb.cs.lmu.edu/biodb/fall2017/index.php/Week_11

Sahara, T., Goda, T., & Ohgiya, S. (2002). Comprehensive expression analysis of time-dependent genetic responses in yeast cells to low temperature. Journal of Biological Chemistry, 277(51), 50015-50021. Doi:

Teixeira, M. C., Monteiro, P. T., Guerreiro, J. F., Gonçalves, J. P., Mira, N. P., dos Santos, S. C., ... & Madeira, S. C. (2013). The YEASTRACT database: an upgraded information system for the analysis of gene and genomic transcription regulation in Saccharomyces cerevisiae. Nucleic acids research, 42(D1), D161-D166.