CSci 3003: Introduction to Computing in Biology

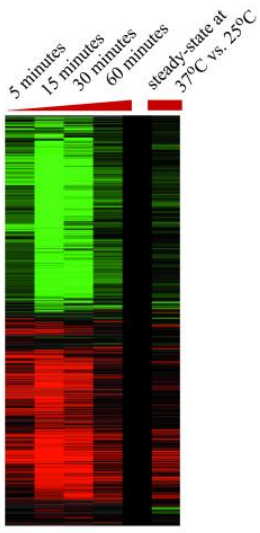
Lab Assignment #8

25 points
Assigned: 11/21/19
Due: Thursday, 12/12/19 (before 11:55pm)

Goals of this lab:

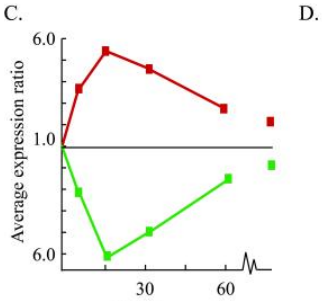
- Practice using R to load/analyze expression data.
- Practice using software tools to explore and interpret a gene expression dataset.

Lab 8: Exploring gene expression signatures through cluster analysis



We will continue focusing on gene expression data but address a different question than we did for the last lab. Instead of looking for genes whose expression is different across two sets of samples (e.g. ER+/ER-), we will look for genes that have similar profiles of expression across large sets of samples. This can be a very powerful technique for discovering the function of uncharacterized genes. For example, two genes that are co-expressed together in response to several different environmental conditions are very likely to perform similar functions in the cell. The data from this lab are taken from the following study, which you can read about here:

Genomic expression programs in the response of yeast cells to environmental changes. Mol Biol Cell. 2000 Dec;11(12):4241-57.
Gasch AP, Spellman PT, Kao CM, Carmel-Harel O, Eisen MB, Storz G, Botstein D, Brown PO.



Part I: Understanding the gene expression study

The first part of this lab will focus on understanding the gene expression study that was done. By following the link above (or simply using Google to find the study), open the paper associated with the data and read the paper's abstract. Briefly answer the following questions about the study:

(1) What was the purpose of the study and what experiments did the authors do to answer the question they were interested in?

(2) What are some examples of different environments or stresses the authors subjected yeast cells to before measuring their gene expression?

(3) Describe one of the significant conclusions of the study.

Part II: Exploring the gene expression data with public software tools

For this part of the lab, we will be using a software tool called Java TreeView, which is a software tool for visualizing gene expression clusters. You can download the tool at:

http://sourceforge.net/projects/jtreeview/files/jtreeview/1.1.6r4/

Download either win (Windows), osx (Mac), or bin (Linux).

(1) If you are on a CSELabs computer, start Java TreeView by first downloading the 'bin' version noted above, and extracting the zip file on your desktop. Then open a Terminal, change (cd) to the TreeView-1.1.6r4-bin folder and type "**`java –jar –Xmx500m TreeView.jar &`**" to start TreeView. (If you are working on your personal machine, you may need to install Java before you can do this).

(2) Start Java Treeview and load the **`Gasch_complete_dataset.cdt`** data file by clicking File->Open. This dataset has already been clustered for you, so you will be able to find patterns of expression.

(3) Describe a few general patterns of gene expression you see across the set of all samples.

(4) Find a small set of genes that has a striking pattern (i.e. a cluster), and look up several of the member genes at http://www.yeastgenome.org. Answer the following questions (be sure to report the set of genes that you've chosen):

   a. Does it make sense that they cluster together (e.g. do they have similar functions)?

   b. What is the pattern of expression that they have in common? For example, under which conditions are they over or under-expressed relative to the reference? Does this make sense given their function?

(5) Find the gene ARG1, which is involved in arginine biosynthesis. Which other genes does it cluster with? What patterns of expression drive their similarity? Why do you think these genes are more/less active under these conditions?

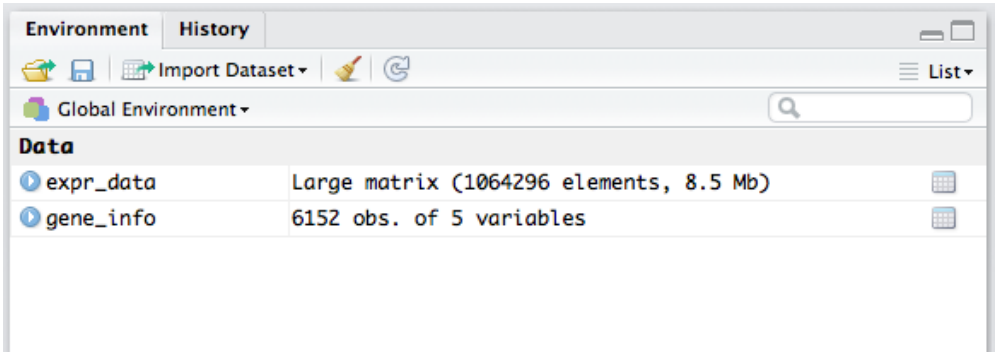
Part III: Clustering gene expression profiles using the k-means algorithm

(1) Download the R code for the k-means algorithm from the Canvas course website.

Open the **kmeans_cluster.R** function in your RStudio editor and study the code. Write pseudo-code that briefly describes what the code is doing (no more than 15-20 lines of pseudo-code are necessary here). What is the stopping criterion for the algorithm?

(2) Download the R version of the Gasch expression dataset from the Canvas website (gasch_expression_data.RData).

Use the **`load( )`** function to load this data into R. You should see the following set of variables in your workspace after loading:



The variable **`expr_data`** is a named matrix whose columns are named after the condition labels and the rows are named after the gene names. **`gene_info`** is a data frame that has one row of information for each gene.

Before clustering, it is often useful to limit the analysis to a subset of genes that shows high variance across the set of conditions studied. In this case, we will restrict our clustering analysis to the 2000 most variable genes. Using R's **`var()`** function (and a call to **`apply()`**), compute variance of each gene across the set of 173 conditions. Then, select the 2000 most variable genes (rows) from the matrix to create a matrix called **`cluster_data`**, which has dimensions of 2000 x 173. This matrix should still have gene names as the row names and conditions as the column names.

(3) Cluster the **`cluster_data`** matrix using the provided **`kmeans_cluster()`** function. Assume the number of clusters is 50 (`k=50`) (***Hint***: the two inputs should be the data matrix and k). The output of this function is a vector of labels (1-50) for each gene indicating which cluster each gene has been assigned to.

(4) Based on the results of #3, answer the following questions:

a. How many iterations did it take for k-means to converge?

b. Add a series of commands to your script that prints the size of each cluster, e.g.

Cluster 1: 50 genes

Cluster 2: 35 genes

What's your biggest and smallest cluster size?

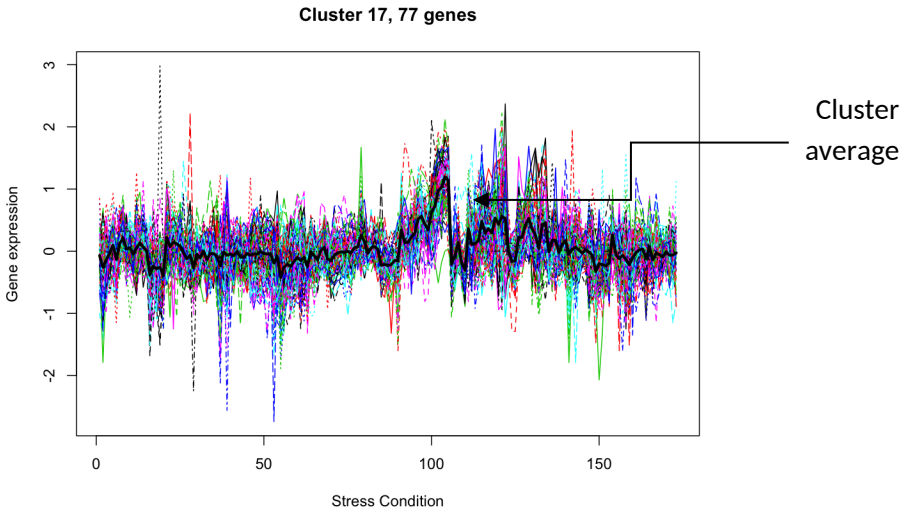
c. For the first 10 clusters, and using a `for` loop, plot 10 different figures showing the expression of the genes within that cluster. Inside the `for` loop, you will likely find the following steps helpful. As an example, let's look at cluster 1:

i. Create a matrix containing only the data for the genes in cluster 1

ii. Compute a vector that is the "average" profile for cluster 1. This represents the average gene expression data across all 173 conditions for the genes in cluster 1.

iii. Using the `matplot()` function, plot all individual gene expression profiles on the same plot. Set `matplot`'s `type` argument such that it plots lines instead of points, and use the `xlab` and `ylab` arguments to give the plot X and Y axis labels. The help for `plot.default()` will be useful

iv. Using the `lines()` function, plot the mean profile as a thick black line on top of the profiles you plotted in part iii. The line width is controlled by the `lwd` argument to `lines()`.



v. Extra credit (1 pt.): using the `` `png()` `` graphics device, export the plots you generate directly to files instead of saving them from the RStudio plot window. Remember, you need to open the graphics device first (`` `png(filename, ...)` ``, followed by drawing the plot, and then closing the graphics device.

d. Pick one of your clusters with an interesting profile (e.g. clear mean expression profile signal in a subset of the conditions). Identify a subset of stress conditions that is either highly over or under-expressed for your chosen cluster. For example, in the case above, you might pick the spike that the arrow points to. (Hint: the column names of **`expr_data`** contain the details for each of the 173 conditions) Under what stress conditions are these genes responding, and in what direction does their expression change?

e. For the cluster you chose, print out the genes that are members of that cluster along with their description and function (from the **`gene_info`** data.frame). Are there any clear patterns in terms of what functions are represented? (Hint: http://www.yeastgenome.org might be useful here)

Part IV: Interpreting your clusters using the Gene Ontology and public tools

For the final section of this lab, we will use a public tool that allows you to systematically evaluate a set of genes (e.g. from a cluster) for enrichment for known functions. This tool is called topGO, and you can install it using the following commands:

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
# Get topGO and yeast annotations
BiocManager::install("topGO")
BiocManager::install('org.Sc.sgd.db')
```

(1) We have provided a script called **topGo_analysis.R,** which demonstrates how to perform GO enrichment on a random set of yeast genes. It downloads and installs all the appropriate packages required for GO enrichment including an annotation file which contains the set of yeast gene annotations.
(2) Examine the code and find where the character vectors **`gene_universe`** and **`interesting_genes`** are created. You will need to assign these to the appropriate values based on your results from part III. The rest of the code for using the topGO package has written for you.
(3) Run the GO enrichment test and extract the results as shown in the script. topGO will check all possible GO terms for enrichment in your cluster and return a list of enriched functions in a data frame.

(4) Investigate the annotations for these GO terms and summarize some of the striking function enrichments you see. Do these functions make sense given the conditions the genes responded to in part III above? Copy and paste the results (or take a screenshot) returned by topGO and include them in your lab report.

Submit to Canvas

When you're finished with the lab, make a report of any questions you answered plus any requested output, and gather the scripts that you modified.

Also, a couple of requests regarding this homework submission:
(1) Please submit any answers to questions in PDF (or *.docx) form and be sure to include your name.
(2) Please do NOT include the original data files with your submission.

Submit your homework file on the Canvas site.