Mining Regulatory Modules from Gene Expression Data

December 3, 2009

1 Introduction

1.1 Proteins, Gene Expression, & Transcription

The fundamental molecular building block of the cell is the *protein.* Proteins are a class of molecule that perform almost every function in the cell: they help to give it structure, they aid the chemical reactions which provide the cell's energy, they form pathways to pass signals inside the cell, and they are the gatekeepers on the cell's surface which regulate what passes through the cell's membrane and what the cell does or doesn't stick to.

The central insight of modern molecular biology is that a kind of molecule called DNA is the blueprint for all of the cell's proteins – DNA comes in large molecules (*chromosomes*), but pieces of the chromosomes (called *genes*) have a chemical sequence which mirror the structure of the proteins. A gene is said to "code for" a protein if the sequence of the gene mirrors the structure of the protein. "Gene expression" is the cellular process by which genes are turned into the proteins for which they code. The process of expression has two distinct steps. First, genes are *transcribed* into an intermediate form (*mRNA*[1]), the carbon copying sheet to DNA's plain paper). Second, the intermediate form is *translated* into the corresponding protein.

Ideally, a biologist interested in measuring the structure and function of a cell (or population of cells) would measure the varying amounts of different proteins present in those cells at different points of the cell's lifecycle, or when the cells were put under different stresses. However, for technical reasons, direct measurements of proteins is difficult – instead, biologists resort to measuring the process of gene expression as a proxy for measuring the total amount of a particular protein. In particular, if a large amount of the intermediate mRNA form of a gene is present in the cells then this is a clue that (perhaps) the corresponding protein is abundant as well.

Because biologists are so interested in measuring the expression of genes (again, as a proxy for proteins), they have developed experimental techniques by which the expression of *all* genes in a genome can be measured simultaneously[2]. For each gene in the genome, the biologist measures the "intensity" of that gene's expression – a real number whose value is related to the abundance of the gene's transcript in a population of cells. A collection of expression experiments forms a data matrix: the rows are *genes*, with one row for each gene in an organism, and the columns are the *experiments.*

[1]the 'm' is for "messenger" and RNA is a class of molecules similar to DNA

[2]Exactly *how* these experiments are performed isn't relevant to this exercise; if you're interested, try searching for the phrases "gene expression microarrays" or "transcript sequencing" in your favorite search engine.

1.2 Co-expression and Gene Modules

The very first thing a biologist will want to discover, given such a matrix of expression values, is which genes are *co-expressed.* Two genes are co-expressed if they share similar patterns of expression across the experiments in a dataset, and co-expression is usually evidence of a higher-order similarity between the genes or their products. Pathways are one such "higher-order" function: a *pathway* is a set of proteins which interact in sequential or regular ways, in order to communicate a message or perform a function within the cell[3]. Since pathways function as a unit, it doesn't do any good for the cell to produce only one protein in the pathway – to activate the pathway, it must produce *all* the proteins in the pathway at the same time. In these cases, we would expect to see co-expression of all the genes coding for the proteins which share a common pathway.

Biologists use this empirical fact (co-expression) to discover new biology (pathways and other higher-order functions). Given one gene, A, that a biologist has previously identified as involved in a pathway or process within the cell, he or she can then identify other genes B, C, and D which are co-expressed with A. The proteins produced by genes B, C, and D are then hypothetically related, functionally, to the protein produced by gene A.

Groups of co-expressed genes are sometimes referred to as *modules* in the biology and bioinformatics literature. A module is a collection of genes which share a common pattern of expression in a common set of experimental conditions. Notice that the genes in a module do not have to be co-expressed in *all* experimental conditions – the function of the module may only be active in some experiments or some kinds of cells. For example, if we have identified an "arsenic response" module of genes, we might only expect that module to show co-expression in those experiments where we have exposed the cells to arsenic first.

1.3 Transcriptional Regulation

But why should we expect to see genes co-expressed at all? How does the cell actually effect a common expression pattern for a set of functionally related genes? The answer lies in the process of *transcriptional regulation.*

Transcription, the key first step in gene expression, doesn't just happen randomly or haphazardly. Each gene is "turned on" by a complex transcriptional machinery which interacts with signals in the DNA sequence of the gene itself. Genes which share a common function (and are part of the same module) will have the same, or similar, versions of these DNA signals – and so they will be affected by the transcriptional machinery in the same way, and be expressed in the same experiments.

Of course, this transcriptional machinery is itself composed of proteins – and those proteins are coded for by genes, which must themselves be expressed! Genes which code for the components of this transcriptional machinery are called *transcription factors*, and we might expect some of these transcription factors to be co-expressed with the genes whose transcription they regulate. For example, if gene A is a transcription factor which activates Gene B[4], then when Gene A is expressed we should see a corresponding increase in the expression of Gene B. This isn't always the case – there can be non-additive effects as well, such as a gene C whose expression requires *both* genes A and B, but not either in isolation. If gene A is constitutively expressed[5], then we might find that gene C is most co-expressed with gene B, and less with gene A.

[3]An example of such a pathway is a *metabolic pathway*, where a sequence of chemical reactions to breakdown a chemical and provide energy for the cell are catalyzed by a set of proteins.

[4]eg. the product of Gene A is a protein which, when present in the cell, forms a machine to "turn on" the expression of Gene B

[5]that is, "always" expressed

Therefore, a tighter classification of genes beyond simple co-expression is *co-regulation* – a co-regulated set of genes is co-expressed because they share a common set of regulatory DNA signals to attract a common set of transcription factors. A module is co-regulated if all the genes in the module share a common set of regulatory transcription factors which coordinately control the module in those experiments where the module is expressed. The discovery of co-regulated modules, and the transcription factors which regulate them, constitute a fundamental goal of the field of functional genomics.

In this exam, we will walk you through a few of the basic tasks involved in identifying gene modules and their common regulators. As you will see, many of the data mining techniques you have seen in class will be immediately applicable to the discovery of functional modules.

1.4 The Goals

For this exercise, we'll be working with measurements of expression in *yeast*, one of the simplest eukaryotes[6] and also one of the most extensively studied. A strain of yeast derived from a beer culture was one of the first organisms to have its complete genome sequence discovered (as a test run, in essence, for the famous Human Genome Project), and the structure of that genome is fairly simple. It contains around 6000 genes with a well-defined naming scheme. Most genes have been discovered to play a role in at least one known function or pathway. The set of transcription factor genes has been known for at least ten years, and there are a wealth of whole-genome datasets made available from publicly-funded research.

We will work, here, with two of those datasets. All datasets are provided as tab-delimited text files, which should be readable in R through the `read.table` command. Each of the files contains an appropriately formatted header line, so the `header` = `TRUE` command should be passed to the `read.table` function, and is tab-separated (requiring an `sep` = "`t`" command to the `read.table` function as well). In each data file, the rows correspond to genes which were measured in those experiments[7], and the columns have arbitrary names indicating the name of the corresponding experiment.

2 An Atlas of Observational Expression Experiments

The first dataset is available in the file `expressdb_cleaned.txt`. This file contains a set of observational expression experiments, compiled from published reports across several different labs and institutions. Each row corresponds to a gene, and each column has a name corresponding to an individual experiment in the dataset. The complete dataset is called "ExpressDB," and was independently curated and redistributed by a lab at Harvard University. For our purposes, you can treat these experiments as "observational" – that is, none of the genes themselves have been directly manipulated. Changes in gene expression between experiments should be attributable either to differences in the experimental conditions that the cells were grown under: different nutrients or different chemical or physical stresses.

[6]The term *eukaryote* is a rough classification, essentially meaning "a complex cell with a nucleus." Bacteria are not eukaryotes, but you and I (and plants, animals, insects, and pretty much every other complex multicellular organism) are.

[7]A word about gene names: "standard" yeast gene names have the format `Yx{RL}###{CW}` . The `x` is a single letter that indicates the chromosome of the gene (chromosome 1 is 'A', chromosome 2 is 'B', etc.), followed by a letter (either `R` or `L`) which indicates which *side* of the chromosome the gene is on. The `###`'s are three digits indicating the *order* of the gene on that side of the chromosome, and the final letter (either `C` or `W`) indicates the *direction* that the gene is point ('C' stands for Crick and 'W' for Watson, the scientists credited with the discovery of the structure of DNA. Therefore, the gene name `YAL010C` would tell a biologist to look for the 10th gene in on the left arm of the first chromosome, which will be pointing in the 'Crick' direction. This was useful in the days before bioinformatics and complete genome maps.

2.1 Calculating Gene Expression Similarity

The first task is to develop a model of *gene expression similarity.* A gene expression similarity function is a method which takes three inputs: two genes, and a set of experiments (possibly the complete set of experiments). It returns a numerical indicating the similarity or difference in expression between the two genes *across the indicated experiments only.*

Notice that these experiments are not necessarily normalized! This means that the intensity values in each experiment (column) are not comparable in a straightforward way between experiments – a value of 1.0 in one experiment may be "equivalent to" a value of 4.5 in another. Investigating the distribution of values within each experiment, and correcting for this difference between experiments, is one of the key challenges in building a reasonable expression similarity function.

One of the simplest tasks you can use a gene expression similarity function for is to rank genes by their similarity with respect to a reference gene, from most similar to least similar. Even before the identification of modules and regulators, biologists will often wish to ask the question "what genes are most similar to Gene X?"

For this problem, you should:

1. Write an R function for the calculation of a gene similarity function. Turn in any R code and corollary data that is used to calculate the function.
2. Discuss what normalization, if any, was necessary in order to create your similarity function.
3. What other properties does your similarity function possess? Is it symmetric (i.e. does it produce the same result when you switch the gene arguments)? What value does it return if you pass it two identical rows (genes) as arguments?
4. Use your similarity function to produce a ranking of genes by their similarity to gene `YPR035W`.
5. Describe (but you do not have to implement) a method for determining a ranking *cutoff* – that is, a similarity threshold within which we could conceivably say that genes are co-expressed with reference `YPR035W`, and outside of which the level of gene expression similarity is too low to determine that the genes are co-expressed.

2.2 Discovering Co-expressed Modules

For the next task, you will use your gene similarity function to discover co-expressed sets of genes in the ExpressDB dataset. As discussed above, a co-expressed module is a collection of genes whose expression is similar across a collection of experiments. In our case, we will ask you to focus (again) on `YPR035W` – we are interested in the module or modules which contain this gene in different subsets of experiments.

1. Write an R function which takes a gene and a set of experiments as input, and produces the module (set of genes) as output which are "co-expressed" with the given gene in the indicated experiments.
2. What properties does your module discovery algorithm have? If you ran it on gene A (producing as output module A), and then chose a gene B not in module A as second input – would the output of the second run, module B, overlap with module A? That is, are the modules you produce *disjoint*?
3. Use the function to find the module containing `YPR035W` in the experiments whose names begin with the prefix `Der_diaux`. Repeat this using the experiments whose names are prefixed with `Spe_cdc15`. Are the genes within the two modules the same? Overlapping? Completely different (aside from `YPR035W` itself)?

4. Discuss (but you do not have to implement) the design of a "module discovery" method – this is a method which takes only a set of experiments as input, and produces *all* the relevant sets of genes which form modules with respect to these experiments.

3 Discovering Regulators from Genetic Knockout Experiment

So far we've looked at "observational" expression data, measurements of gene expression that don't involve any direct manipulations of the genes themselves. We've used this data to discover co-expressed collections of genes, some of which might reasonably be taken to form modules with a common function. However, as discussed above, co-expression is too loose for many of our purposes – we would like to resolve our co-expressed modules into *co-regulated* modules. These are collections of genes which are grouped together by virtue of both a common expression pattern *and* a common set of regulatory transcription factors.

We could start with a list of the transcription factors themselves, but even this is not enough: if gene A is a transcription factor, and is co-expressed with gene B, how do we decide whether the co-expression is due to a common set of regulators, or because gene A *regulates* gene B? One step towards the answer might be given by a perturbational dataset – we could manipulate gene A, and see if gene B's expression undergoes a corresponding change.

In this final section of the exam, we will look at just such a perturbational expression dataset. Unfortunately, biology is a messy business – we don't have the experimental ability to finely manipulate the expression of genes. We can, however, manipulate them in crude ways: we can "knock them out." A "knock out" strain of yeast is a variant which has been created with an *edited* version of its genome – it is *missing* a particular gene. If this doesn't kill the yeast outright, then the resulting yeast cells are guaranteed to have *zero* expression of the knocked out genes (because they're just not there).

In a paper from 2007, Zhanzhi Hu and his collaborators from the University of Texas published a "knock-out" expression atlas for every (200+) transcription factor in the yeast genome. That means that, for each transcription factor, they systematically created a strain which knocked out *just* that gene and then measured the resulting expression of all the remaining genes. The data for these experiments is available in the file `HuIyer_TFKO_expression.txt`, but a quick word of description is required these value. The value for each gene in the matrix is actual a relative intensity[8] comparing the expression level of the gene in the knockout strain against a "background" (unmodified) strain measured simultaneously. A negative value means the expression level is much *lower* in the knocked out strain than in the background, while a positive value means the expression level is higher[9]. As with the first experiments, however, you cannot assume that the values are *normalized* or comparable across experiments.

3.1 Discovering Co-regulated Modules

1. Write an R function to discover the regulators of a module – this is a method which takes as input a module (a set of genes), and produces a list of all regulators (these can be experiment names from the Hu/Iyer dataset) which are regulators of the genes in that module.

2. Discuss the characteristics of this function – are any of the regulators discovered for the module part of the module itself? (Note that *self*-regulation, where a transcription factor regulates both a set of

[8]Specifically, it is the logarithm of a ratio.

[9]Higher expression levels are perfectly reasonable, since some transcription factors regulate their target genes by *repressing* them – akin to turning off a light switch. If you remove the regulator which has kept its targets off, you would expect to see the expression levels of the target go up. We won't explore this wrinkle in this exercise.

target genes and itself, is a well-known biological phenomenon.)

3. Discover the regulators of one of the modules that you discovered for gene XXX, above.
4. Given that regulators can regulate each other as well as other genes, discuss (but do not implement) the method by which you might try to discover the *regulatory network* of the cell – this is the complete set of regulatory interactions between transcription factor genes and their targets.