SCIE2100 Practical 9, Week 12

Aim

In this practical you will be re-introduced to the Gene Ontology (GO) and GO enrichment testing. Specifically, you will be discovering enrichment of roles in groups of genes that share some property that previous bioinformatics methods have helped identify. After this practical you will have an appreciation of GO and how it can be used to interpret results of bioinformatics analyses. You will also gain an understanding of the Fisher Exact test—a statistical test that is commonly used in bioinformatics.

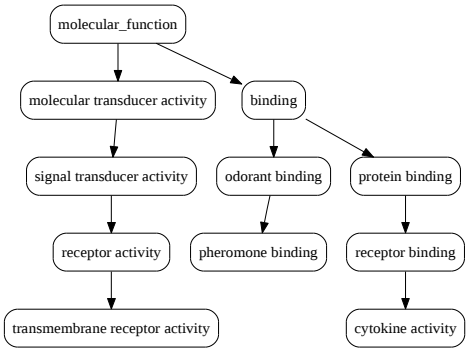
Files

For this week's practical you will need this guide, python files `go.py`, `genome.py` and `stats.py` (amongst others) and data files `yeast_promoters.fa`, `yeast_go.py` and `GDS3198.soft`; this gene expression data was obtained from the Gene Expression Omnibus (GEO; http://www.ncbi.nlm.nih.gov/geo/).

Gene Ontology

GO annotations provide a controlled vocabulary for discussing the localisation (cellular component), function (molecular function) and process involvement (biological process) of gene products (e.g., proteins). GO annotations allow us to, for example, search for genes or proteins via their known functional roles in an automated way. GO annotations are available for many model organisms, including human and yeast. We will stick to the latter in this practical, but rest assured that the same analysis could be done across many species.

GO annotations are organised into a structure called a directed acyclic graph (DAG); this structure is similar to a tree, except in that parents can have multiple children. The graph on the right is a tiny sub-graph of GO terms; you can see how more general terms (closer to the top) are linked to more specific 'child' terms. In this graph, the coloured items are terms that were found to be overrepresented in their association within a set of genes of interest.



Statistical analysis of functional roles

We will be revisiting two types of methods from previous practicals, namely motif searching and finding differential gene expression. Both these methods can be used to identify a set of genes that are special in some way. For instance, we will look for genes that have promoters with a certain transcription factor-binding motif. Or, genes that are significantly up- or down-regulated after perturbing the ability of a transcription factor.

After this preliminary identification of a “gene set”, we will use GO to see if there is anything special about this set, relative to a set of neutral/typical/normal genes. This information can be hugely important to the biologist—so what we deem “special” needs to be couched in terms of scientifically robust statistics…

Abf1 is a general regulatory factor that contributes to transcriptional activation of a large number of genes, as well as to replication, silencing and telomere structure in yeast. In spite of its widespread use for transcription, the specific role of its targets is poorly characterized, and the range of targets genome-wide is not adequately understood. This practical will highlight how so-called gene set enrichment analysis can be used to do just that.

The Fisher Exact Test (FET) provides a way to quantify statistical significance of an association between two distinct properties. Many popular gene set enrichment tools use the FET, by checking if genes in a set of interest (referred to as positives) have members that are assigned a given property, to an extent that cannot be explained by chance; either too many or too few. Below is an example of a 2x2 contingency table.

	Has property	Does not have property	*Row total*
In gene set of interest	a	B	*a+b*
Not in gene set of interest	c	D	*c+d*
Column total	*a+c*	*b+d*	*a+b+c+d*

From this matrix a p-value can be calculated for the association. More information about Fishers Exact Test can be found online at http://en.wikipedia.org/wiki/Fisher's_exact_test. Please read this at least cursorily before you continue.

The most common “property” in gene set enrichment analysis is a specified GO term; call it *T*. A gene *G* would either be *positive* or not, and either be annotated with *T*, or not.

1. Lets assume we perform an experiment on 25 genes in yeast. We find 8 that respond to a treatment *X*. Those 8 are our positives, namely YPL106C, YOL081W, YOR027W, YOR299W, YNL006W, YNL007C, YLL039C and YLR216C. Consequently, 14 genes will make up our negative set.

   Determine the support for this proposition: *X* is indicative of a function known as ‘chaperone regulator activity’ known as GO:0030188 by the Gene Ontology. We know that the following 11 genes are annotated with GO:0030188: YER048C, YIL016W, YLR090W, YOR027W, YMR161W, YNL064C, YNL281W, YDR214W, YPL106C, YNL007C and YNL227C. In fact, we see *two* of them amongst our negatives.

   Hint: `stats.py` contains an implementation of FET. If you have the numbers in the table above, you can do the following.

   ```
   >>> import stats
   >>> stats.getFETpval(a, b, c, d, left=False)
   ```

`left=False` means that the test looks at *over*-representation of the term in the positives. If you change it to `left=True` it looks at *under*-representation. Submit: The statistical support for the above proposition according to FET. Is it fair game to say that *X* is indicative of chaperone regulator activity; yes or no?

Identification of gene sets

Data set 1: which genes have promoters with binding sites for Abf1?

Transcription factors tend to bind just upstream of gene start sites. Indeed in yeast (*Saccharomyces cerevisiae*) they tend to bind within 800 base pairs but not always. Looking further afield may dilute our analysis so we will stick to this. RSAT (http://rsat.ulb.ac.be/) provides an easy way to obtain promoter sequences. We have already downloaded sequence for all yeast genes: `yeast_promoters.fa`. (If you wish to generate it yourself, see end of this document.)

Next we find what Abf1 likes to bind to. JASPAR (http://jaspar.genereg.net/) is a database of motifs representing transcription factor binding sites. Figure shows what the Abf1 motif looks like.

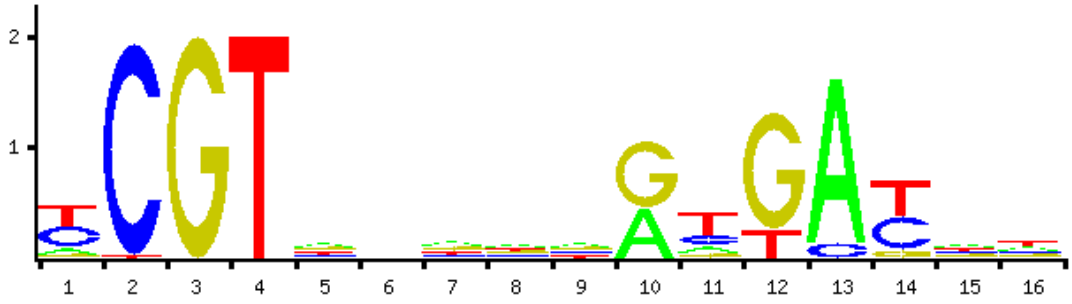


Figure : Abf1 binding motif.

`prob.py` has method `readMultiCount` that can read the JASPAR file, which you also find in the practical's file directory: `abf1.jaspar`. `readMultiCount` returns a list of probability distributions that you can pass to the `PWM` class in `sequence.py`.

2. Load the promoter sequences. How many yeast promoters are there? Verify that this number is biologically sensible and cite your source. Submit: the number and source.

3. Create a PWM for Abf1. Verify that the probabilities in your new PWM correspond to those visualised in the above sequence logo. Submit: print the probabilities in the 10th column/position and explain why (a) you only see G and A in the logo, and (b) why the total height is around 1 bit.

4. Use the Abf1 binding motif to score all yeast promoters. Assign to each promoter the maximum score attained over all positions. Don't print them all… no need to submit anything yet.

   Hint 1: Take a look at `PWM.maxscore` which scores a sequence using a PWM.

Hint 2: Put your results in a map, keyed by the gene name, e.g.

```
>>> bind_map = {}
>>> for s in yeast_prom: # yeast_prom is an array of sequences
        bind_map[s.name] = abf1_pwm.maxscore(s)[0] # save score only
```

5. Using so-called ChIP-seq experiments, by the most conservative estimates, MacIsaac and colleagues identified at least 200 targets of Abf1. Plot the score distribution over all promoters as a histogram. Here's how you can do it:

```
>>> scores = []
>>> for s in bind_map.keys():
       if bind_map[s] != None:
              scores.append(bind_map[s])

>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> hist, bins = np.histogram(scores, bins=50)
>>> width = 0.7 * (bins[1] - bins[0])
>>> center = (bins[:-1] + bins[1:]) / 2
>>> plt.bar(center, hist, align='center', width=width)
>>> plt.show()
```

   Use the histogram to decide on a threshold, that identifies at least 50 targets but maybe more. Submit: histogram, justified threshold and the exact number of promoters that meet this threshold. List the 50 target genes.

Data set 2: which genes change as a result of disabling Abf1?

Yarragudi et al. (2007) used microarrays to examine the contribution of Abf1 to transcription genome-wide, by using mutants that dissociate from their binding sites at 37C. They performed a control experiment with wild-type Abf1 and then with mutant Abf1, in three replicates. The expression data are available from the GEO, with accession GDS3198, though you should use the file provided on Blackboard. We will identify genes that vary in their expression from control to mutant. (We performed a similar analysis in an earlier practical; if you need a refresher on gene expression analysis, you may wish to reread that document.)

6. Load the gene expression data. How many probes are there in your file? How many unique gene names are there?

   Hint 1: remember that a probe is not a gene, and that a probe *may* map to one gene, but that a gene may have zero, one or more probes associated with it.

   Hint 2: Note that `GDS3198.soft` lists the standard yeast identifiers in the second column, thus you could read this file and index by this identifier `id_column = 1`, e.g.

```
import genome as ge
g = ge.readGEOFile('GDS3198.soft', id_column = 1)
```

   If `id_column = 0` the probe identifier is used, and you would get a different number of entries in the map.

Submit: Numbers of probes and genes.

7. Obtain the set of differentially expressed genes with support, from the data set.

   Hint 1: Inspect the file in particular the header to understand what experiments are recorded, what the measurements really are, and in what order they appear. You will notice that the values are raw counts. Hence, you need to worry about normalisation.

   Hint 2: To assess support for calling difference in expression, you can first pair your replicates, i.e. replicate 1 for wild-type vs. replicate 1 for mutant Abf1, replicate 2s against one another and same with replicate 3s. To compare pairs of experiments, it is common to use ratios, and take logarithms of them.

   I averaged the log-ratios, like so:

```
g = ge.readGEOFile('GDS3198.soft', id_column = 1)
meanfold = {}
for gene in g.genes:
    profile = g.getGenes(gene)
    meanfold[gene] = (math.log(profile[0] / profile[3]) +
math.log(profile[1] / profile[4]) + math.log(profile[2] / profile[5]))
/ 3
```

   Plot the distribution of the log-ratios, like so:

```
import matplotlib.pyplot as plt

scores = [y for y in meanfold.values() if not np.isnan(y)]
hist, bins = np.histogram(scores, bins=50)
width = 0.7 * (bins[1] - bins[0])
center = (bins[:-1] + bins[1:]) / 2
plt.bar(center, hist, align='center', width=width)
plt.show()
```

   Does it look normal? Would a Z-score be able to assign a level of significance to the difference in expression of a single gene?

   Next, print the names and log-ratios of the 100 probes that have the lowest expression, and the 100 probes that have the highest expression in wild type cells:

```
result = sorted(meanfold.items(), key=lambda v: v[1])
print '========== Wildtype may down-regulate =========='
for r in result[0:100]:
    print r[0], r[1]
print '========== Wildtype may up-regulate =========='
for r in result[-1:-100:-1]:
    print r[0], r[1]
```

   Submit: a description of the data transformations and filtering you did, even if you followed the recipe above.

8. Select about 50 Abf1 target genes. You will note that many probes printed above will have curious names, because they map poorly to known protein

coding genes. Remove them, and select probes with sensible gene names and high differential expression (negative or positive).
Submit: the gene list.

Evaluating enrichment in gene sets

As hinted earlier, a common way of analysing function using GO terms is to look at statistical over-representation of GO terms that are associated with a set of genes, e.g., differentially expressed genes in a condition, or genes that share a binding motif in their promoter. Each of those genes may have several GO terms associated. The question then is: is a particular GO term unusually common within this set? This question is very similar to the one asked initially, but now we want to consider *all* GO terms. Luckily, we can do this one term at a time, and identify only those that meet a statistical threshold.

In order to establish statistical overrepresentation, you need to have both a foreground set (i.e., your positives or genes of interest) and a background set (i.e., a set of genes that can establish the general occurrence of GO terms). In our case, the set of differentially expressed genes is our positive set. In some cases the background can have a significant impact on the enrichment p-value, but we will ignore this for now. We will assume that all genes that we know of form our background.

Familiarise yourself with `go.py`. Explore in particular what the section at the end does (designated `__main__`) and run it. You should recognise the names and term, since we used them in exercise 1. You can model how you solve the problems below on this code. (Notice that the actual GO database is imported from `yeast_go.py` when you construct a `GODB` class instance. If you want to use this code for other species, follow the instructions at the back, and convert the data to Python maps as exemplified in `yeast_go.py`.)

9. Check out `get_GO_term_overrepresentation`. You will notice that the p-value that FET calculates is modified before returned, then called E-value. Recall that BLAST determined an E-value. Why does the method return this E-value and not the p-value? Submit: explanation to E-value in this context.

10. Find GO terms that are over-represented in the Abf1 promoter set (from exercise 5). Submit: GO terms (if any) that have an E-value smaller than 1.0.

11. Find GO terms that are over-represented in the Abf1 differential expression set (from exercise 8). Submit: GO terms (if any) that have an E-value smaller than 1.0.

Interpreting results

GO terms can be a useful resource for identifying commonalities, but can be challenging to interpret. It is generally a good idea to inspect the results of a GO analysis to see if you can identify common themes, and also to identify any suspect terms.

12. Look at your two sets of significantly over-represented GO terms. Are there apparent common themes in your term set? Are there any terms that look out of place? You need to make use of the method `get_GO_description` to find out what the term means.

    Given that this data set is associated with Abf1, which of the top results look reasonable? You may need to do some quick research on your terms in order to answer this question.

    Submit: common themes, and oddballs in each of your result sets from exercises 10 and 11; biological insights relating to Abf1.

References

Yarragudi et al. Genome-wide analysis of transcriptional dependence and probable target sites for Abf1 and Rap1 in Saccharomyces cerevisiae. Nucleic Acids Res. 35(1) 2007.

MacIsaac KD, Wang T, Gordon DB, Gifford DK, Stormo GD, Fraenkel E. An improved map of conserved regulatory sites for Saccharomyces cerevisiae. *BMC Bioinformatics*. 2006 Mar 7;7:113. http://www.ncbi.nlm.nih.gov/pubmed/16522208. (Note: Supplementary data including transcription factor binding sites and target genes is available at http://fraenkel.mit.edu/improved_map/).

Data sets

How do I generate the promoter set?
Go to the RSAT web page, and click “retrieve sequence”. Select organism; “all” genes; sequence type “upstream”; “default” size is good. Press “GO” to download, and save as FASTA. The label “gene name” will give you standard yeast gene names that will match GDS3198’s second identifier.

How do I find the original Gene Ontology data?
The maps used by `go.py` (which you can find in `yeast_go.py`) are a combination of two separate databases, namely the ontology file, and the yeast annotation file. The ontology file defines the three ontologies (molecular function, biological process and cellular component) for all species. You can download the latest here: http://www.geneontology.org/GO.downloads.ontology.shtml. The annotation file is specific to yeast, and there would be a file like this for each species supported by GO. You find them here: http://www.geneontology.org/GO.downloads.annotations.shtml.