Identifying transcription factor targets by microarray analysis of mutant plants

Introduction

Cytokinins are plant hormones that regulate nearly every aspect of plant development and physiology. Cytokinins activate a signal transduction pathway in plant cells that causes phosphorylation and activation of transcription factors called type B ARRs (for "authentic response regulators"). Once converted to their active form, type B ARR proteins bind to the promoters of downstream genes and activate their transcription. Some of the most dramatically induced downstream genes include called type A ARR genes. Mutations in type A ARRs make plant cells more sensitive to cytokinin and thus play a role in shutting down the pathway following activation. Most plant species contain several genes encoding type B ARRs, and genetic studies have shown that their functions are partially redundant.

One commonly performed assay in cytokinin research is called a root growth inhibition assay, in which a cytokinin hormone (such as trans-zeatin or benzyl adenine (BA), dissolved in DMSO, an organic solvent) is sprayed onto seven or ten day old seedlings sprouted on agar plates. After a few days (typically three or five) the investigator measures the amount of root growth and compares it to the growth of similarly grown, untreated seedlings. Application of cytokinin causes a dramatic reduction in root growth in the treated versus control plants, and this reduction in root growth increases with increasing amounts of applied cytokinin. However, plants bearing mutations in genes required for perception or transmission of the cytokinin signal are less sensitive to the hormone and so their roots continue to grow at a rate similar to untreated plants.

In today's assignment, you'll use tools from Bioconductor to process and analyze data from a DNA microarray experiment that assayed gene expression in wild-type and mutant Arabidopsis thaliana seedlings undergoing treatment with trans-zeatin, a kind of cytokinin. The mutant plants in this experiment were double mutants in two genes: ARR10 and ARR12, both encoding type B ARRs.

To understand the experiment, its rationale, and how the original investigators analyzed their data, you should read the attached article, titled Type-B ARR transcription factors, ARR10 and ARR12, are implicated in cytokinin-mediated regulation of protoxylem differentiation in the roots of Arabidopsis thaliana. Pay particularly close attention to how the investigators analyzed their microarray data. Note that they used Genespring, a commercial analysis package from Agilent.

When the paper was published (2007) methods for analysis of microarray data were not as well-developed as they are now, and so it is possible that by using today's methods, you could uncover new (and important) information about how cytokinin signaling works by re-analyzing the array data produced in the study. In general, this is nearly always the case for data coming from high-throughput, highly quantitative techniques such as RNA-Seq or DNA microarrays. As statistical methods improve, we often are able to go back to older research, re-analyze the data, and learn new information.

Description of the experiment

The experiment involves two experimental factors: treatment type (S or T) and genotype (WT or Mu). Plants bearing mutations in two type B ARRs (ARR10 and ARR12), along with wild-type controls, were treated with transzeatin for one hour and then harvested along with untreated controls of both genotypes. Three "batches" (or replicates) of each treatment group were then harvested and processed for analysis using the ATH1 microarray platform (GPL198) from Affymetrix.

In this lab, you'll use tools from the Bioconductor package to process the data from the experiment, identify differentially expressed genes, and also attempt to answer the question: which genes respond differently to cytokinin in the mutant in comparison to wildtype? Such genes, if we can find them, may represent potential downstream targets regulated by the ARR10 and ARR12 transcription factors.

Setting up

Download the data

Download the CEL files for samples from GEO accession GSE20232. Go to: http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE20232 and follow the Supplemental File link at the end of the page. Unpack the file using tar.

```
$ mkdir GSE20232
$ cd GSE20232
$ wget
ftp://ftp.ncbi.nlm.nih.gov/pub/geo/DATA/supplementary/series/GSE20232/GSE20232_RAW.
tar
$ tar -xf GSE20232_RAW.tar
$ ls
GSE20232_RAW.tar  GSM507132.CEL.gz  GSM507137.CEL.gz
GSM507128.CEL.gz  GSM507133.CEL.gz  GSM507138.CEL.gz
GSM507129.CEL.gz  GSM507134.CEL.gz  GSM507139.CEL.gz
GSM507130_truncated.CEL.gz GSM507135.CEL.gz
GSM507131.CEL.gz  GSM507136_truncated.CEL.gz
```

Note two CEL file names contain the word "truncated." When the data were submitted to the repository, part of the files were lost, rendering them useless for re-analysis.

Download the meta-data describing the experiment.

Each experiment captured in the GEO has an associated meta-data file that describes the experiment. It's called a "Series Matrix File." Download it, uncompress it, and open it in a text editor.

```
$ wget
ftp://ftp.ncbi.nlm.nih.gov/pub/geo/DATA/SeriesMatrix/GSE20232/GSE20232_series_matri
x.txt.gz
$ gunzip GSE20232_series_matrix.txt.gz
$ emacs GSE20232_series_matrix.txt
```

Scroll to the section that starts Sample_title. Note that each sample in the experiment as a title (such as 2-1_CK-treatment-wildtype_Rep1_ATH1) and that each sample has a corresponding GEO sample accession, such as GSM507128. The original CEL files are named for the corresponding GEO sample accessions.

These are listed in order, allowing you map CEL file names onto sample names and thus understand the structure of the experiment.

QUESTION What does the meta-data file say about the truncated CEL files?

Launch R and load Bioconductor packages

For this step, you may need to install Bioconductor packages affy, limma, and a few others, depending on how the lab computers are configured.

Launch R from UNIX:

```
$ R

R version 2.15.0 (2012-03-30)
Copyright (C) 2012 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-apple-darwin9.8.0/x86_64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

Ideally, you are running R version 2.15.0 or higher. (Lesser version may also work.)

Load required Bioconductor packages

Depending on how your computer is configured, you may already have the required Bioconductor packages already installed. From inside R, try this:

```
> library(affy)
Loading required package: BiocGenerics

Attaching package: 'BiocGenerics'

The following object(s) are masked from 'package:stats':

    xtabs

The following object(s) are masked from 'package:base':

    anyDuplicated, cbind, colnames, duplicated, eval, Filter, Find,
    get, intersect, lapply, Map, mapply, mget, order, paste, pmax,
    pmax.int, pmin, pmin.int, Position, rbind, Reduce, rep.int,
    rownames, sapply, setdiff, table, tapply, union, unique

Loading required package: Biobase
Welcome to Bioconductor

    Vignettes contain introductory material; view with
    'browseVignettes()'. To cite Bioconductor, see
    'citation("Biobase")', and for packages 'citation("pkgname")'.
```

If you instead see an error like

```
Error in library(affy) : there is no package called 'affy'
```

then you need to install the affy Bioconductor package and the other libraries it depends on.

Take a moment to visit the Web page for the affy package:

http://www.bioconductor.org/packages/release/bioc/html/affy.html

Note that it depends on number of additional packages and that several other packages depend on it.

To install affy, load the biocLite.R script into your R environment and run the biocLite function:

```
> source("http://bioconductor.org/biocLite.R")
> biocLite("affy")
```

Try loading affy again after installing the affy package and its dependencies.

```
> library(affy)
```

Repeat the above process for the limma package.

Process the CEL files

Read the data

Create experimental meta-data file

This experiment involves two experimental factors: treatment and genotype. To start, create a data frame that contains information describing each sample and the values for each factor for each sample.

Create a comma-separated file named samples.txt that contains the following sample information:

```
id,genotype,treatment,filename
WT.U1,WT,U,GSM507131.CEL.gz
WT.U2,WT,U,GSM507132.CEL.gz
WT.U3,WT,U,GSM507133.CEL.gz
WT.S1,WT,S,GSM507128.CEL.gz
WT.S2,WT,S,GSM507129.CEL.gz
MT.U1,Mu,U,GSM507137.CEL.gz
MT.U2,Mu,U,GSM507138.CEL.gz
MT.U3,Mu,U,GSM507139.CEL.gz
MT.S1,Mu,S,GSM507134.CEL.gz
MT.S2,Mu,S,GSM507135.CEL.gz
```

Save it in the same directory where you launched R.

⚠ The final line of samples.txt requires a newline character.

Read the data into a data structure with associated meta-data

```
> pd = read.AnnotatedDataFrame("samples.txt",header=TRUE,sep=",",row.names=1)
> mydata = ReadAffy(filenames=pd$filename,phenoData=pd,verbose=TRUE)
> sampleNames(mydata)=row.names(pData(pd))
```

You can usually just type a variable name to find out what type of object it represents, its class, and other useful information. Try it now:

```
> mydata
> pd
```

QUESTION: What kind of object is mydata? What kind of object is pd?

To find out what kinds of methods you can apply to an object, use R's help function. Try this:

```
>help(AffyBatch)
```

Scroll to the section titled "methods." These are methods you can apply to any object of type AffyBatch. You'll use the "image" and "hist" functions to check array quality, as described in the next section.

Check data quality

Use the image function to check for defects in the arrays.

```
image(mydata)
```

You should look for arrays that have obvious defects such as unusually dim or bright regions.

Use the hist function to check that the distribution of probe values from each array is similar.

```
> hist(mydata)
```

You should see ten curves that are more or less the same shape.

QUESTION: Do you observe any problem arrays? If yes, remove them by repeating the previous steps and editing the samples.txt file.

Convert probe intensity values to normalized expression values

There are many methods for converting probe intensity values to probe set level expression values; the two most commonly used are MAS5 and RMA.

MAS5 stands for microarray suite 5 and was developed at Affymetrix. It averages the difference between perfect match and their corresponding mismatch probes and requires a subsequent normalization step. RMA stands for robust multichip averaging and incorporates a probe level normalization step prior to generating probe set level expression values. Both methods are properly called probe set summarization methods because they convert individual probe intensity values read from the scanner into summarized probe set level values representing expression values for individual target transcripts.

In this lab, we'll use RMA.

Use RMA to convert probe intensity values to probe set expression values.

Use RMA (with default options) to create a new object (eset) containing normalized expression values for individual probe sets:

```
> eset = rma(mydata)
```

QUESTION: What type of object is eset?

Differential expression analysis

In this step, we'll use the limma library to identify three sets of genes that differentially expressed.

We'll use limma to find

- genes that are differentially expressed between treated and untreated wildtype plants (test wild-type treated versus wild-type control)
- genes that are differentially expressed between treated and untreated mutant plants (test mutant treated versus mutant control)
- genes that are differentially expressed between the first two groups of differentially expressed genes (test mutant versus wildtype treated versus control)

The third group requires some explanation. As discussed in the Introduction, we hypothesize that the ARR10 and ARR12 genes play a role in activating changes in gene expression in response to cytokinin. If so, then when we compare gene expression changes observed in the wild type plants to gene expression changes in the *arr10 arr12* double mutant plants, we expect to find that some genes that respond in the wild type fail to respond in the same way in the mutant. In other words, we expect to observe that expression of many genes will increase or decrease in the wildtype and that some of these will do nothing in the mutant or will increase or decrease to lesser degree.

If our hypothesis is correct and if our experiment has sufficient statistical power (which it may not) then the third comparison should identify genes whose response to cytokinin depends on proper functioning of ARR10 and ARR12.

Create a design matrix

The first step in using limma to perform these three classes of test is to create a design matrix, a data structure that describes the structure of our experiment.

To start, create factor objects that represent the different experimental groups.

```
> TS=paste(pd$genotype,pd$treatment,sep=".")
> TS = factor(TS, levels = c("WT.U", "WT.S", "Mu.U", "Mu.S"))
```

The first command creates TS, a simple list of strings labeling each sample with a label representing its experimental group. The second command converts TS into a list of factor objects.

Now, use TS to create a design matrix:

```
> design=model.matrix(~0 + TS)
> colnames(design)=levels(TS)
> design
   WT.U WT.S Mu.U Mu.S
1     0    1    0    0
2     0    1    0    0
3     1    0    0    0
4     1    0    0    0
5     1    0    0    0
6     0    0    0    1
7     0    0    0    1
8     0    0    1    0
9     0    0    1    0
10    0    0    1    0
attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$TS
[1] "contr.treatment"
```

The design object captures information about how the experiment is structured and it should match the structure of the expression data set. Each row corresponds to a sample and each column indicates whether or not the sample belongs to the group indicated by the column label.

Fit a linear model to the data.

Next, use lmFit (from the limma package) to fit a linear model to the data. Explaining exactly what this means is beyond the scope of this class. For now, think of it as a way to model the relationships between sample type (membership in a particular group) and expression of individual genes in those samples.

```
> fit = lmFit(eset,design)
> fit
```

Note however when you type "fit," R will print a lot of information about its contents.

Note that "fit" and "design" are both S3 objects (not S4 objects), meaning you can't as easily find out what methods apply to them. However, you can access their components using the $ operator.

For example, to get a matrix of the log-transformed average expression values for every probe set in each group, you can do this:

```
> head(fit$coefficients)
             WT.U     WT.S     Mu.U     Mu.S
244901_at 5.832888 5.956477 5.820174 5.804942
244902_at 6.163383 6.118981 6.153163 5.903767
244903_at 6.802523 6.976344 6.972242 6.901941
244904_at 6.163504 6.044170 6.358280 6.296296
244905_at 4.619321 4.586435 4.680115 4.689613
244906_at 7.133443 6.935662 6.807688 6.682159
```

The log2 fold-change between WT.U and WT.S is given by

```
> fit$coefficients[,1]-fit$coefficients[,2]
```

Create contrasts matrix and test for differential gene expression

```
> cont.matrix= makeContrasts(
+   WT.SvsU = WT.S - WT.U, # minus means: compare these groups
+   Mu.SvsU = Mu.S - Mu.U,
+   Diff = (Mu.S - Mu.U) - (WT.S - WT.U), # compare the difference of differences
+   levels = design) # provides the info needed to know what WT.S, etc means
> fit2 = contrasts.fit(fit,cont.matrix)
> fit2 = eBayes(fit2) # allows variance borrowing
```

The last command requires some explanation. Recall that the Introduction mentioned that statistical methods for analyzing microarray have been improving?

One classical problem with microarray experiments has been that investigators rarely can afford to perform more than three biological replicates per group. However, to perform a statistical test, one needs to estimate variance, and variance is harder to estimate when there are not many replicates. However, the variances for indvidual genes is probably very similar, and so statisticians have developed a technique that estimates variance for individual genes using all of the genes. This technique is called "shrinkage" and is implemented in the eBayes method implements. Improving the estimation of the variance has the happy effect of increasing the experiment's power to detect real changes.

Now use fit2 to determine the overlap in terms of differential expression between the three sets of tests.

```
> results = decideTests(fit2,p.value=0.01)
> vennDiagram(results)
```

The diagram shows the number of DE genes (probe sets) in each comparison, including the genes that are differentially regulated depending on the genotype.

QUESTION: How many genes are differentially regulated depending on the genotype. (The Diff group)?

QUESTION: What do you think about this result? Feel free to discuss it with other students in the class in person or on the message board.

Retrieve gene annotations

The next step in the analysis is to associate gene information (names of genes and descriptions of their functions) to probe set ids. For this, we'll use a set of probe set to gene annotations from the Arabidopsis Information Resource (TAIR). The TAIR Web site is located at www.arabidopsis.org.

```
>
f="ftp://ftp.arabidopsis.org/Microarrays/Affymetrix/affy_ATH1_array_elements-2010-1
2-20.txt"
> annots = read.delim(f, na.strings = "", fill=TRUE, header=T, sep="\t")
> sm.annots=annots[,c(1,5,6)]
> names(sm.annots)[1]='ID'
> head(sm.annots)
```

Now we have a data frame that contains probe set ids, gene ids, and gene descriptions.

Select top genes from each comparison

For this, use topTable, which extracts rows from fit2 with adjusted p value below a cutoff, where adjusted means: increased to account for multiple hypothesis testing. In this case, we will use a cutoff of 0.01.

```
> N=dim(eset)[1]
> top.WT.SvsU = topTable(fit2,coef=1,number=N,p.value=0.01)
> top.Mu.SvsU = topTable(fit2,coef=2,number=N,p.value=0.01)
> top.Diff = topTable(fit2,coef=3,number=N,p.value=0.01)
```

Merge annotations with top lists

For this, use a function called merge, which allows you to do combine two tables of data using a common column. In this case, both tables have a common column: ID, which contains a probe set id.

```
> top.Diff=merge(sm.annots,top.Diff)
> top.Mu.SvsU=merge(sm.annots,top.Mu.SvsU)
> top.WT.SvsU=merge(sm.annots,top.WT.SvsU)
```

Write the data to a file.

For each list of DE genes, write the data to a file:

```
write.table(top.Diff,file='top.Diff.txt',row.names=FALSE,sep='\t')
write.table(top.WT.SvsU,file='top.WT.SvsU.txt',row.names=FALSE,sep='\t')
write.table(top.Mu.SvsU,file='top.Mt.SvsU.txt',row.names=FALSE,sep='\t')
```

Interpret your data

Open the files in a spreadsheet program or text editor and scan the list of genes, starting with top.Mu.SvsU.txt.

You should see several genes that are annotated as having a role in cytokinin signaling, including:

- CKX3
- ARR9
- ARR5
- CKX4
- ARR7
- ARR15

QUESTION: What are the fold-change of these genes as reported in top.Mu.SvsU.txt? What is the fold-change of these genes in top.WT.SvsU?

QUESTION: How do you interpret this result?

Turn in your work

Use svn to commit a plain text file containing answers to the questions.

Optional bonus questions

Read the section of the paper in which the authors discussed their observation that many genes that were up-regulated in the wild-type plants were also up-regulated in the mutant plants, but not as much. Do your results agree?

To find out, use merge to combine top.Mu.SvsU and top.WT.SvsU.

```
> both=merge(top.Mu.SvsU,top.WT.SvsU,by=c('ID'))
> both = both[,c('ID','locus.x','description.x','logFC.x','logFC.y','adj.P.Val.x',
      'adj.P.Val.y')]
> names(both)=c('probeset','agi','description','MutantLogFC','WildtypeLogFC',
      'Mutant.p','Wildtype.p')
```

These steps create a new data structure (both) that is the intersection of genes found to be differentially expressed in both the mutant and the wildtype.

Now, let's find out: How many genes expression changed in the same direction in both the mutant and the wildtype in response to BA?

For this, multiple the log2 fold-changes and then ask how many of the products are non-zero. This works because down-regulated genes have negative log fold-changes and up-regulated genes have positive fold-changes. The product of a negative and a positive number is negative, but the product of a negative and negative or a positive and a positive is positive.

```
> v = (both$MutantLogFC * both$MutantLogFC > 0)
> sum(v)
```

QUESTION: What was the result?

Now find out how many genes changed in the same direction but not as much in the mutant as compared to the wildtype.

```
> v = (both$MutantLogFC * both$MutantLogFC > 0) &
          (abs(both$WildtypeLogFC) - abs(both$MutantLogFC) > 0)
> sum(v)
```

QUESTION: How do the results compare to what was reported in the paper?