Microarray Data Analysis

Article *in* Methods in molecular biology (Clifton, N.J.) · January 2011

DOI: 10.1007/978-1-60761-682-5_3 · Source: PubMed

2 authors:

Saroj Mohapatra
National Institute of Biomedical Genomics
68 PUBLICATIONS **742** CITATIONS

SEE PROFILE

Arjun Krishnan
Princeton University
66 PUBLICATIONS **3,945** CITATIONS

SEE PROFILE

Chapter 3

Microarray Data Analysis

Saroj K. Mohapatra and Arjun Krishnan

Abstract

Gene expression profiling has revolutionized functional genomics research by providing a quick handle on all the transcriptional changes that occur in the cell in response to internal or external perturbations or developmental programs. Microarrays have become the most popular technology for recording gene expression profiles. This chapter describes all the necessary steps for analyzing Affymetrix microarray data using the open-source statistical tools (R and bioconductor). The reader is walked through all the basic steps of data analysis: reading raw data, assessing quality, preprocessing/normalization, discovery of differentially expressed genes, comparison of gene lists, functional enrichment analysis, and saving results to files for future reference. Some familiarity with computer is assumed. This chapter is self-contained with installation instructions for R and bioconductor packages along with links to downloadable data and code for reproducing the examples.

Key words: Gene expression, Statistical analysis, Bioinformatics, Differential expression, Gene Ontology

1. Introduction

Transcriptional regulation is a complex process that plays a pivotal role in reprogramming cellular states in response to internal or external changes that arise because of progress through different phases of growth and recycling or perturbations. Hence, measuring and analyzing this regulation could lead to important discoveries regarding the regulatory molecules and the downstream mediators of the response. For over a decade, many high-throughput technologies have been developed for profiling gene expression by monitoring genome-wide mRNA levels, the most prominent among them being the microarray technology.

Andy Pereira (ed.), *Plant Reverse Genetics: Methods and Protocols*, Methods in Molecular Biology, vol. 678,
DOI 10.1007/978-1-60761-682-5_3,

1.1. Basic Principle of Microarray Technology and the Biological Data that It Offers

Microarray is a small chip that contains thousands of probes fixed to its surface, which can hybridize with fluorescently labeled RNA samples (the targets). Hybridization intensities, represented by the amount of fluorescent emission, give an estimate of the relative amounts of the different transcripts that are present in the RNA sample. Many different microarray platforms exist that differ in array fabrication and dye selection. Affymetrix Genechips (1) are high-density oligonucleotide microarrays with 25-nt probes that contain multiple probes per gene (together called a probeset) for most genes in the genome. This platform has gained enormous popularity because of reasonable accuracy and coverage.

A single microarray experimental assay, like most high-throughput technologies, records the transcript levels of all the genes in a given condition, for a given cell type or mixture, at a given time. Comparison of the transcript levels of genes (represented by virtual values) to a reference assay and between different assays is hence important for the interpretation of changes in gene expression. Here, the experimental design heavily influences the quality, quantity, and even the validity of the information obtained. Much care is therefore needed in developing a sound design taking into account several factors, including the quality of samples, the amount of replication and pooling.

Assuming that we have a completed microarray experiment and the resulting gene expression profile data, the following sections bring forth concepts underlying the various steps involved in statistical analysis of this data toward biological inference.

1.2. Statistical Analysis of Microarray Data

1.2.1. Quality Assessment and Control

Issues with RNA extraction (sample quality and amount of starting material), labeling, scanning, or even array manufacture can affect the quality of the microarrays. Visual inspection and creating diagnostic plots can help assess and possibly filter the data. Several of the following factors are taken into account when assessing the quality of microarray chips: average background (expected to be similar across all chips), scale factors (expected to be within threefold of one another), percentage of the number of genes called present (expected to be similar across all chips), and ratio of expression of 3′ probes to 5′ probes of "housekeeping" genes β-actin and GAPDH (acceptable if less than 3 and 1.25, respectively).

The reader is referred to the "Guidelines for Assessing Data Quality" section in the Affymetrix data analysis manual (available at http://www.affymetrix.com/support/downloads/manuals/data_analysis_fundamentals_manual.pdf) for further information on quality control.

1.2.2. Preprocessing

Preprocessing is the process of extracting and transforming the raw fluorescence intensities into a signal normalized for experimental

errors and biological variation. The first step in preprocessing is background subtraction (removal of background noise) followed by the normalization to remove systematic sources of variation in the measured intensities due to wide variety of factors. These include early factors, such as print-tip differences (when multiple printing pins are used to print each chip), other spatial effects, and the quality of the microarray printing, and later factors such as quality of the mRNA used, separate reverse transcription and labeling, different dye labeling efficiencies, and different scanning parameters. Procedures, such as quantile normalization (2), make the assumption that the different biological samples have roughly the same distribution of RNA abundance, and transform the intensities so that the bulk of the intensity distribution is the same for all assays in an experiment, typically with some differences in the distribution tails (which might reflect actual biological differences). For Affymetrix data, this method is applied at the probe-level before the prove-level intensities are summarized into a probeset-level value. Methods such as Robust Multiarray Averaging (RMA) perform background correction, quantile normalization, and summarization (3).

1.2.3. Comparison Between Groups to Get Differentially Expressed Genes

Genes that respond to the condition under scrutiny can be discovered by comparing their expression levels across a pair of groups, treatment versus control, later- versus earlier-time-point, infected versus uninfected, etc. Genes that show consistent expression across replicates within a group and difference between groups can be uncovered by employing a statistical test. But, to deal with experimental design issues, the main one being the usual small group size (small number of replicates), modified versions of conventional statistical methods (e.g., moderated *t*-test) have to be used, as implemented in the analysis package limma (4). All the genes can then be ranked based on the significance of differential expression and genes that have a p-value < 0.05 (say) can be declared as the set of differentially expressed genes (DEGs). However, performing multiple statistical tests (one for each of the thousands of genes on the array) at the same time raises the alarm of genes having a p-value < 0.05 just by chance. Methods that control the false discovery rate (FDR) (5), the proportion of false positives among the genes declared as significantly different, have proven to be very useful in overcoming the multiple hypothesis testing problem (see ref. 6 for very good discussion of related issues).

1.2.4. Functional Analysis of Differentially Expressed Genes

Further biological knowledge can be extracted from the list of DEGs by portioning the genes into coherent groups that are known to perform biological processes or participate in a biochemical pathway. The significance of any such functional category (e.g., terms in the Gene Ontology (GO) (7)) that a subset

of DEGs belong to can be tested by calculating the probability of observing as many or more genes (among the DEGs) belonging to that category given the total number of DEGs, the number of genes in the genome annotated with that category and the total number of genes in the genome. The hypergeometric test takes these quantities to calculate a *p*-value for the enrichment of genes belonging to a category among the DEGs.

Recent reviews contain more rigorous discussions of other potential analysis methods and issues (8–10).

Here, we concentrate on one simple series of steps for the analysis of gene expression data produced using a standard single-channel microarray platform (Affymetrix GeneChip). Similar steps can be extended to the data from other platforms. We will be using bioconductor tools (11) within R statistical computing environment (12) for the analysis. It is assumed that RNA extracted from some samples has been hybridized onto microarrays and the machine-output data are available for analysis. The example dataset used in this chapter contains the raw data files (CEL format; see Note 1) corresponding to three abiotic stress treatments in *Arabidopsis* along with their respective controls (13). mRNA from shoot samples harvested 30 min post-treatment was used for gene expression profiling using the *Arabidopsis* ATH1 Genome Array (see Note 2). Through a series of steps, we take you from reading in the raw data files to finding DEGs, followed by some basic functional analysis of the genes.

2. Materials

R (http://www.r-project.org/) is an excellent free software environment for statistical computing and graphics. With its wide variety of statistical and graphical techniques, and high extensibility, it has become the standard for analysis and representation of data, including biological data. Bioconductor (http://www.bioconductor.org/) is a free, open source, and open development software project for the analysis and comprehension of genomic data. It is based on R and its components are distributed as R packages, several of which are written for the analysis of microarray data. To keep with the big wave in microarray analysis, we present all the steps needed to perform the analysis entirely within R.

1. Install R on your local machine. Go to http://www.r-project.org, under "Download, Packages" section on the left, click CRAN, select the mirror site nearest to you, and then based on your operating system download the corresponding binary and install it based on the instructions provided therein.
2. In addition to the default packages in R, we need a basic set of packages from bioconductor, the *Arabidopsis* array database

and the GOstats package for functional enrichment analysis (14). Start R (see Note 3). From within R, run the following commands (without the initial ">"):

```
> source("http://www.bioconductor.org/
  biocLite.R")
> biocLite()
> biocLite("ath1121501.db")
```

For the GOstats package, go to http://bioconductor.org/packages/devel/bioc/html/GOstats.html, download the package corresponding to your operating system, and place it in any folder. If you are using Linux/UNIX/Mac do 'R CMD INSTALL GOstats_2.11.0.tar.gz' outside of the R session (in your terminal); if you are using Windows, within an R session go to the "Packages" menu, select 'Install from a local zip file', and select the downloaded file (see Note 4).
Now, load the libraries.

```
> library(affy)
> library(limma)
> library(simpleaffy)
> library(ath1121501.db)
> library(GOstats)
```

3. In your machine, create a directory for this analysis and within it, create another directory called *celfiles* for the raw data files. Then, download the CEL files from TAIR (15) into this directory. Go to http://www.arabidopsis.org/servlets/TairObject?type=hyb_descr_collection&id=ID, replacing "ID" with each of 1007966668, 1007966553, and 1007966888 for the drought, cold, and salt data, respectively. For each stress, download the four CEL files corresponding to shoot tissue: two replicates of the control samples and two replicates of 30 min post-treatment samples.
4. In the current directory (one level above "celfiles"), create a file called *Target.txt* in the format presented in Fig. 1 that

Name	Celfile	Group
Drought.Control.1	DROUGHT_CONTROL_30MIN_SHOOT_REP1.cel	Drought.Control
Drought.Control.2	DROUGHT_CONTROL_30MIN_SHOOT_REP2.cel	Drought.Control
Drought.Stress.1	DROUGHT_30MIN_SHOOT_REP1.cel	Drought.Stress
Drought.Stress.2	DROUGHT_30MIN_SHOOT_REP2.cel	Drought.Stress
Salt.Control.1	SALT_CONTROL_30MIN_SHOOT_REP1.cel	Salt.Control
Salt.Control.2	SALT_CONTROL_30MIN_SHOOT_REP2.cel	Salt.Control
Salt.Stress.1	SALT_30MIN_SHOOT_REP1.cel	Salt.Stress
Salt.Stress.2	SALT_30MIN_SHOOT_REP2.cel	Salt.Stress
Cold.Control.1	COLD_CONTROL_30MIN_SHOOT_REP1.cel	Cold.Control
Cold.Control.2	COLD_CONTROL_30MIN_SHOOT_REP2.cel	Cold.Control
Cold.Stress.1	COLD_30MIN_SHOOT_REP1.cel	Cold.Stress
Cold.Stress.2	COLD_30MIN_SHOOT_REP2.cel	Cold.Stress

Fig. 1. Sample annotation of the RNA samples used for hybridizing the microarrays. The format resembles that of the *Targets* file.

contains the annotation of the samples in 3 (tab-separated) columns: *Name* for a short sample name, *Celfile* for the name of the raw data file corresponding to the sample, and *Group* for the type of stress and treatment. The group information is especially important for creating contrasts, i.e., pairs of sample groups that are compared with each other.

Now, we are ready to begin the analysis.

3. Methods

3.1. Reading Raw Data

1. Navigate to the current directory (called *Example_dataset* here) using the following command, replacing the path here with the path of the folder in your machine.

```
> setwd("E:/Microarray_Analysis/
  Example_dataset")
```

2. Read the sample annotation from the *Targets.txt* file to an R variable called *targets*.

```
> targets = readTargets("Targets.txt")
```

3. Next, create an R object called *phenoData* that stores the metadata information about the samples. This is important for organizing the sample-level information along with gene-level expression data obtained from CEL files.

```
> rownames(targets) = targets[,1]
> nlev = as.numeric(apply(targets, 2,
  function(x)
+ nlevels(as.factor(x))))
> metadata = data.frame(labelDescription =
+ paste(colnames(targets), ": ", nlev, "
  level",
+ ifelse(nlev==1,"","s"), sep=""),
+ row.names=colnames(targets))
> phenoData = new("AnnotatedDataFrame",
+ data=targets, varMetadata=metadata)
```

4. Read the data from CEL files (along with sample metadata from *phenoData*) to create an object *dat*, which contains the raw probe-level data.

```
> dat = ReadAffy(sampleNames =
  targets$Name,
+ filenames = targets$Celfile,
+ phenoData = phenoData, celfile.path =
  "celfiles")
```

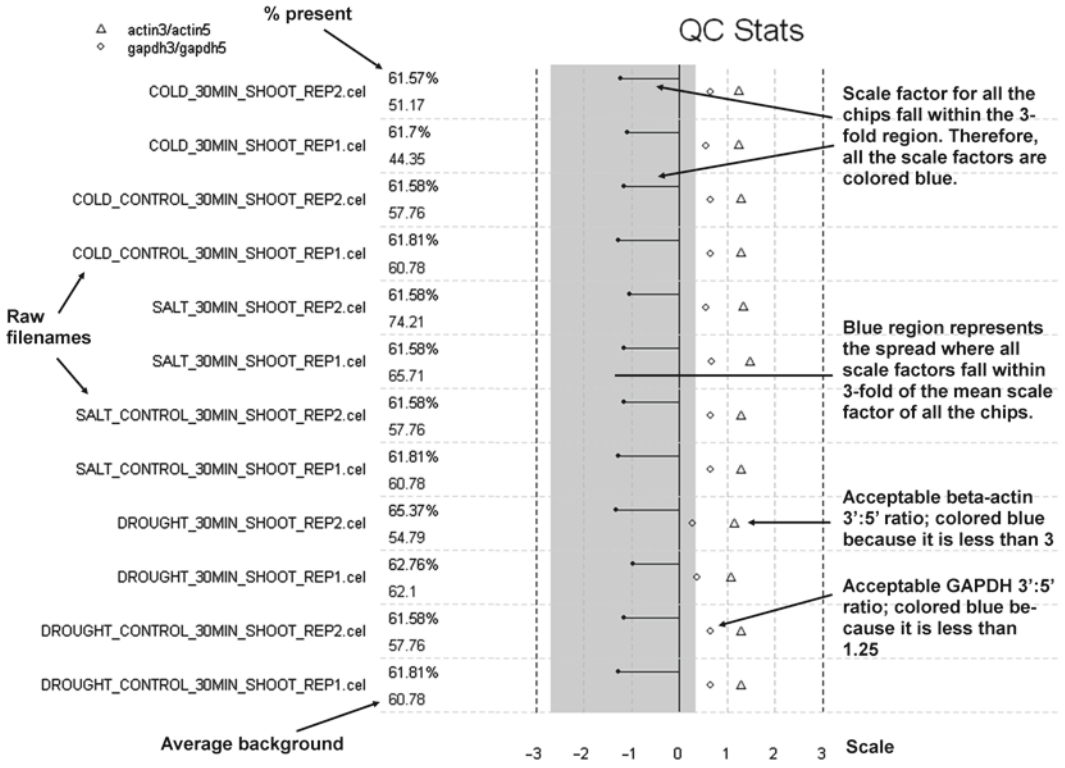

Fig. 2. QC summary statistics. All the arrays appear to be of good quality. Different metrics and their expected and observed quantities are annotated.

3.2. Quality Assessment

1. To assess the quality of the samples, make use of the function *qc* in package *simpleaffy* (16) for this purpose (Fig. 2) which computes a number of statistics based on recommendations from the array manufacturer Affymetrix.

```
> myqc = qc(dat)
> plot(myqc)
```

3.3. Preprocessing

1. Next, probeset-level data needs to be extracted by applying a normalization algorithm. Here, RMA is used (see Note 5). The normalized data object *eset* is used in further steps below.

```
> eset = rma(dat, verbose = FALSE)
```

2. The effect of quantile normalization (part of RMA) – to bring about similar distribution of gene expression data for each individual array – can be seen by plotting a boxplot for the data before and after normalization (Fig. 3; see Note 6).

```
> par(mfrow=c(1,2), mar=c(12,5,3,3))
> mycols = rep(c("lightgreen", "orange",
  "green", "red",
```

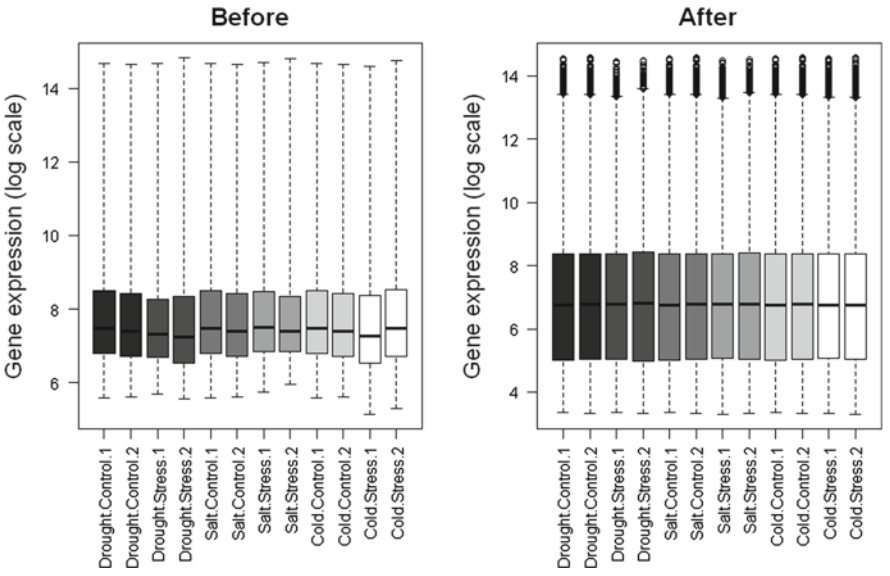

Fig. 3. Quantile normalization imposes the same empirical distribution of intensities on each array. Plots show distribution of gene expression in individual samples before (*left*) and after (*right*) normalization. Each *colored box* corresponds to one array, with same color for each sample group. The *middle line* corresponds to median. Observe that the *boxes* are more aligned at their medians on the *right.*

```
+ "blue", "pink"), each=2)
> boxplot(dat, col=mycols, main="Before",
+ ylab="Gene expression (log scale)", cex.
  lab=1.5,
+ names=dat$Name, las=2, cex.main=2)
> boxplot(data.frame(exprs(eset)),
  col=mycols,
+ main="After", ylab="Gene expression (log
  scale)",
+ cex.lab=1.5, names=eset$Name, las=2, cex.
  main=2)
```

3.4. Discovering Interesting Genes

To find interesting genes that respond to stress, we do the following four comparisons for all three types of stress: drought, salt, and cold. This can be performed using moderated *t*-test available in the package *limma*. There are four steps: (1) define the design matrix and contrast matrix, (2) fit a linear model through each gene, (3) perform empirical Bayes moderation of the standard errors, and (4) list the most interesting genes. Practically, it amounts to applying four functions from the package *limma*: *model.matrix* (*+makeContrasts*)→*lmFit*(*+contrasts.fit*)→*eBayes*→*topTable*.

1. Creating a design matrix and contrast matrix. The design matrix indicates which RNA samples have been applied to

```
> print(design)
   Cold.Control Cold.Stress Drought.Control Drought.Stress Salt.Control Salt.Stress
1             0           0               1              0            0           0
2             0           0               1              0            0           0
3             0           0               0              1            0           0
4             0           0               0              1            0           0
5             0           0               0              0            1           0
6             0           0               0              0            1           0
7             0           0               0              0            0           1
8             0           0               0              0            0           1
9             1           0               0              0            0           0
10            1           0               0              0            0           0
11            0           1               0              0            0           0
12            0           1               0              0            0           0
attr(,"assign")
[1] 1 1 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$grp
[1] "contr.treatment"
```

```
> print(cont.diff)
                 Contrasts
Levels            Cold Drought Salt
  Cold.Control      -1       0    0
  Cold.Stress        1       0    0
  Drought.Control    0      -1    0
  Drought.Stress     0       1    0
  Salt.Control       0       0   -1
  Salt.Stress        0       0    1
```

Fig. 4. *Top*: The design matrix indicates the RNA sample hybridized to each array. *Bottom*: The contrast matrix indicates the comparisons that are of interest.

each array. It is created by applying the function *model.matrix* on the *Group* information in sample metadata. Each row of the design matrix corresponds to an array and each column corresponds to a coefficient used to describe the RNA source (Fig. 4 *top*). Since we are dealing with Affymetrix arrays, the number of coefficients (columns in the design matrix) is same as the number of distinct RNA sources.

```
> grp = factor(eset$Group)
> design = model.matrix(~0+grp)
> colnames(design) = c("Cold.Control",
  "Cold.Stress",
+ "Drought.Control", "Drought.Stress",
  "Salt.Control",
+ "Salt.Stress")
> print(design)
```

We have an experiment with six groups that allows $6 \times 5 = 30$ possible pair-wise comparisons. However, we are interested in only a subset of these, i.e., three contrasts of Stress versus Control. Thus, we need to create the contrast matrix which indicates the comparisons that are of interest. This can be achieved by applying the function *makeContrasts* on the design matrix (Fig. 4 *bottom*).

```
> cont.diff = makeContrasts(
+ Cold = Cold.Stress-Cold.Control,
+ Drought = Drought.Stress-Drought.Control,
```

```
+ Salt = Salt.Stress-Salt.Control,
+ levels = design)
> print(cont.diff)
```

2. Fitting a linear model. The systematic part of the data can be fully modeled by the linear model specified by the design matrix, and the initial coefficients can be compared in selected ways (as specified in the contrast matrix).

```
> fit = lmFit(eset, design)
> fit2 = contrasts.fit(fit, cont.diff)
```

3. Empirical Bayes moderation. For assessing differential expression, the empirical Bayes method can then be used to moderate the standard errors of the estimated log ratios.

```
> fit2 = eBayes(fit2)
```

3.5. Exploring the List of DEGs

1. The list of top genes that are differentially expressed by drought stress, for e.g., (at default adjusted *p*-value cutoff of 0.05) can be obtained using the function *topTable* (Fig. 5; see Note 7).

```
> options(digits = 3)
> topTable(fit2, coef = "Drought")
```

The column *logFC* reports the fold change (log scale) in gene expression by drought stress. Because fold change is a ratio, it sometimes helps to know the absolute gene expression level: column *AveExpr* (very low values should not be trusted too much). The other important column is *adj.P.val*; a value of less than 0.05 means that the fold change is statistically significant.

2. While *topTable* does return the list of most interesting genes, it does not inform about all the genes that are differentially expressed, nor how many of such genes exist for the contrast. Another function *decideTests* from package *limma* needs to be

```
> topTable(fit2, coef = "Drought")
              ID logFC AveExpr     t  P.Value adj.P.Val     B
13239 258139_at  2.69    7.15  26.1 4.13e-09  6.40e-05 10.76
2308  247208_at  1.79    5.99  24.2 7.60e-09  6.40e-05 10.36
20166 265066_at -1.83   10.95 -23.4 1.01e-08  6.40e-05 10.17
9094  253994_at  2.08    9.07  23.0 1.12e-08  6.40e-05 10.09
17548 262448_at  2.24    4.72  21.9 1.72e-08  7.44e-05  9.78
14532 259432_at  2.42    5.44  21.5 1.96e-08  7.44e-05  9.68
6256  251156_at  2.47    4.52  20.9 2.44e-08  7.46e-05  9.52
7231  252131_at  3.76    6.10  20.7 2.62e-08  7.46e-05  9.47
2379  247279_at  1.56   10.09  19.6 4.04e-08  1.02e-04  9.13
19106 264006_at  1.35    9.89  18.8 5.57e-08  1.27e-04  8.87
```

Fig. 5. List of top genes that are differentially expressed by drought stress. The important columns are: *logFC* – fold change (log scale) in gene expression; *AveExpr* – absolute gene expression level; and, *adj.P.val* – *p*-value after adjusting for multiple hypothesis testing.

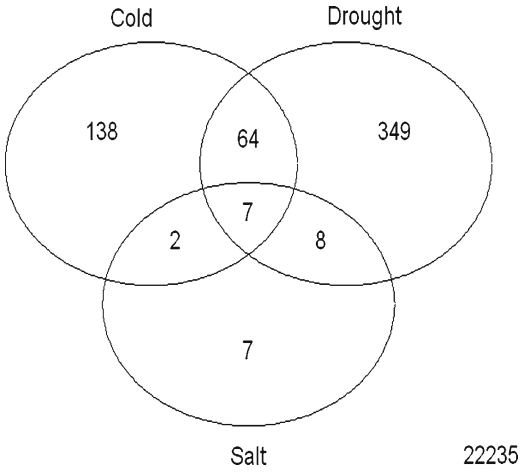

Fig. 6. Venn diagram of the number of genes differentially expressed in the three conditions. Each *circle* corresponds to a set of genes differentially expressed (either up- or downregulated) by particular stress in a statistically significant manner. The name above the *circle* refers to the type of stress. The overlapping area between *two circles* contains the number of genes that are modulated by two types of stress. The number in the *center* (7) refers to the number of genes modulated by all three types of stress. The number on the *lower right corner* indicates the genes unaffected by any stress.

used to obtain the different lists of DEGs, which can then be compared to each other using a using Venn diagram (Fig. 6).

```
> result = decideTests(fit2, lfc=1)
> vennDiagram(result)
```

3. The Venn diagram shows the distribution of stress-regulated genes ($p < 0.05$; absolute fold change >2). To identify genes of interest from here, we need to work on *result*. Let us look at the first five rows in this object (Fig. 7 *top*).

```
> print(result[1:5, ])
```

4. As we can see, each row corresponds to one gene, each column to one stress, and the numbers suggest the direction of differential expression: no change (0), upregulation (1) or downregulation (−1) by stress. Based on this, we can extract the genes in different regions of the Venn diagram and display their fold changes from the data available in the *fit2* object (Fig. 7 *bottom*):

```
> drought.genes = names(which(result
  [, "Drought"] != 0))
> cold.genes = names(which(result[, "Cold"]
  != 0))
> salt.genes = names(which(result[, "Salt"]
  != 0))
> common.genes = intersect(intersect
  (drought.genes,
```

```
> print(result[1:5, ])
           Contrasts
            Cold Drought Salt
  244901_at    0       0    0
  244902_at    0       0    0
  244903_at    0       1    0
  244904_at    0       0    0
  244905_at    0       0    0
> print(lfc)
           Contrasts
             Cold Drought  Salt
  245176_at -1.19   -1.95 -1.92
  245334_at -1.09   -1.07 -1.10
  252607_at  2.21    1.90  1.06
  257076_at -2.36   -2.85 -2.27
  261658_at -1.81   -2.06 -1.50
  263421_at -1.64   -2.65 -2.24
  264857_at -1.02   -1.13 -1.22
> print(mget(common.genes[3], ath1121501GENENAME))
$`252607_at`
[1] "xyloglucan endo-transglycosylase" "xyloglucan endo-transglycosylase"
```

Fig. 7. *Top*: A subset of the object *result* that contains the data for differential expression in the three stresses. Each row corresponds to one gene, each column to one stress, and the numbers suggest the direction of differential expression: no change (0), upregulation (1) or downregulation (–1) by stress. *Bottom*: List of seven genes commonly regulated by the three stresses, and the name of the gene consistently upregulated.

```
+ salt.genes), cold.genes)
> lfc = fit2$coef[common.genes, ]
> print(lfc)
```

The third gene is upregulated by all the stresses. The name of the gene corresponding to that probeset id can be found by referring to the *ath1121501 database* that contains mappings between several types of identifiers and annotations (see Notes 8 and 9).

```
> print(mget(common.genes[3],
  ath1121501GENENAME)
```

5. All the results from linear modeling can be saved to a tab-delimited file that contains fold changes and *p*-values for all probe set ids thus:

```
> write.fit(fit2, file = "limmaResult.txt",
  adjust = "BH")
```

The file *limmaResults.txt* will be created in your current folder and can be explored using a text editor or programs like Excel. The file contains the log fold changes (*Coef.Cold*, *Coef.Drought*, *Coef.Salt*), adjusted *p*-values (*p.value.adj.Cold*, *p.value.adj.Drought*, *p.value.adj.Salt*) as well as other estimates. This file is suitable for manipulating the results outside of R environment and doing any further analysis.

3.6. GO Enrichment Analysis

1. For this analysis, let us concentrate on a particular stress, drought, and identify the genes upregulated by this stress (see Note 10).

```
> drought.genes.up = names(which(result[,
  "Drought"] > 0))
```

2. We can use the genes from this object as our selected list of genes to analyze the enrichment of functional categories.

```
> selectedGenes = unlist(mget(drought.
  genes.up,
+ ath1121501ACCNUM))
```

3. For the enrichment analysis, we also need to define the list of all genes that are in the array as the universe (total number of genes in the genome).

```
> allGenes = featureNames(eset)
> geneUniverse = unlist(mget(allGenes,
  ath1121501ACCNUM))
```

4. Now, set the parameters for the hypergeometric function in the object *params* (see Note 11) and perform the enrichment (overrepresentation) analysis of GO biological process (BP) categories using the function *hyperGTest* (see Note 12).

```
> params = new("GOHyperGParams", geneIds =
  selected Genes,
+ universeGeneIds = geneUniverse,
+ annotation = "ath1121501.db", ontology =
  "BP",
+ pvalueCutoff = 0.001, conditional =
  FALSE,
+ testDirection = "over")
> hgOver = hyperGTest(params)
```

The results of the analysis can be exported as an HTML report that contains the various statistics of each enriched GO term, which are hyperlinked to the GO database (http://www.geneontology.org/).

```
> htmlReport(hgOver, file = "report_hgOver.
  html")
```

Although you could go to your current directory and click this report to open it in your browser, R provides a more elegant way of doing this:

```
> browseURL("file:///E:/Microarray_Analysis/
  Example_dataset/report_hgOver.html",
+ browser="C:/Program Files/Mozilla Firefox/
  firefox.exe")
```

As always, replace the file path and Firefox's (or your favorite browser's) location with the ones in your machine.

5. The report contains the GO Biological Process identifiers (*GOBPID*), their *p*-values (*Pvalue*) that indicate the level of significance, odds ratio (*OddsRatio*) that is an indicator

of the level of enrichment of genes within the list as against the universe, expected number of genes annotated with that term (*ExpCount*) in the list based on the sizes of the list and the universe, actual number of genes in the list annotated with that term (*Count*), number of genes in the universe annotated with that term (*Size*), and the GO term (*Term*).

One of the significantly enriched terms that is of interest is "transcription, DNA-dependent" (GO:0006351) that annotates transcription factors (TFs). There are 25 drought-regulated TFs in the list. These genes can be obtained for scrutiny thus:

```
> drought.tf.genes.up = intersect
  (drought.genes.up,
+ unlist(get("GO:0006355",
  ath1121501GO2ALLPROBES)))
```

Finally, any set (or all) of these genes can be listed with their gene ids and annotations using the following command (Fig. 8).

```
> for(i in 23:25) { y=unlist(mget(drought.
  tf.genes.up[i],
+ ath1121501GENENAME));
+ x=unlist(mget(drought.tf.genes.up[i],
+ ath1121501ACCNUM));
+ cat(strwrap(unlist(c(x[1],y[1])),width=70),
  sep="\t");
+ cat(sep="\n")}
```

```
> for(i in 23:25) { y=unlist(mget(drought.tf.genes.up[i], ath1121501GENENAME));
+     x=unlist(mget(drought.tf.genes.up[i], ath1121501ACCNUM));
+     cat(strwrap(unlist(c(x[1],y[1])), width=70),sep="\n"); cat(sep="\n") }
AT2G46680
encodes a putative transcription factor that contains a homeodomain
closely linked to a leucine zipper motif. Transcript is detected in
all tissues examined.  Is transcriptionally regulated in an
ABA-dependent manner and may act in a signal transduction pathway
which mediates a drought response.

AT2G44840
encodes a member of the ERF (ethylene response factor) subfamily B-3
of ERF/AP2 transcription factor family. The protein contains one AP2
domain. There are 18 members in this subfamily including ATERF-1,
ATERF-2, AND ATERF-5.

AT2G38470
Member of the plant WRKY transcription factor family.  Regulates the
antagonistic relationship between defense pathways mediating
responses to P. syringae and necrotrophic fungal pathogens.
```

Fig. 8. Three of the 25 TFs upregulated by drought stress.

4. Notes

1. CEL file format description: The CEL file includes for each probe on the array an intensity value, standard deviation of the intensity, the number of pixels used to calculate the intensity value, a flag to indicate an outlier as calculated by the algorithm, and a user defined flag indicating the feature should be excluded from future analysis. Generally, this file is used as an input for further analysis and supporting annotation files are either already present as part of the system or automatically downloaded.
2. The Arabidopsis ATH1 Genome Array contains >22,500 probe sets representing ~21,000 genes.
3. R typically starts in a console with a prompt (">"). Therefore, R code presented here starts with the prompt symbol (">") and is written in a different font to distinguish it from the rest of the text. In some cases, the command extends beyond the line and the symbol "+" is used to indicate continuation from the previous line. To reproduce the analysis presented here, please strip off any ">" or trailing "+" and issue the same command in your R console. Alternatively, you could use the R code provided along with the raw data. You should be able to get the same (or similar) result as presented here.
4. GOstats (version 2.10.0) that you could get from within R (using biocLite("GOstats")) does not support GO enrichment analysis for *Arabidopsis*, while this functionality has been added to the development version of the package, which is GOstats_2.11.0. However, please pay attention to the fact that these development version packages might be unsupported, not fully tested and even buggy. So if you then encounter problems, be warned.
5. There are other options for the normalization algorithm such as mas5 (17) and gcrma (18) that you can try in the place of rma. Also, try the command with option `verbose=TRUE` to see the steps printed out as the method progresses.
6. Use the menu "Windows" to alternate between the window displaying the figure and the R console.
7. If you wish to look into more genes and filter them for (adjusted) *p*-value and fold-change, try the following command: `topTable(fit2, coef = "Drought", number = 100, p.value = 0.01)`. *coef* here refers to the contrast name. For finding out the top genes for other contrasts, just change the parameter from *Drought* to *Cold* or *Salt*. Now, it is a good time to show the command that opens up a help/manual

page on any function: `?topTable`. Simply prefix the function by a question mark. If you simply wish to find out all the functions that have anything to do with "table" within R from all the loaded packages, type: `??table`.

8. You can find out about the other types of identifier and annotation mappings available in this database using the following command: `help(package="ath1121501.db")`.
9. This gene is a xyloglucan endo-transglycosylase that plays a role in plant cell wall biogenesis. Members of this family have been reported to be upregulated and assume a protective role in cell wall modification in response to diverse environmental stresses and hormone stimuli (see ref. 7 and the references therein).
10. Type `length(drought.genes.up)` to find the number of genes.
11. Duplicate identifiers, which come about because of multiple probesets mapping to the same gene, will be removed from both the selected and the universe lists.
12. Type `print(hgOver)` to find out the summary of the enrichment test.

References

1. Lockhart, D., Dong, H., Byrne, M., Follettie, M., Gallo, M., Chee, M., et al. (1996) Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nat Biotechnol.* **14**: 1675–1680.
2. Bolstad, B. M., Irizarry, R. A., Åstrand, M., and Speed, T. P. (2003) A comparison of normalization methods for high-density oligonucleotide array data based on variance and bias. *Bioinformatics.* **19**: 185–193.
3. Irizarry, R. A., Hobbs, B., Collin, F., Beazer-Barclay, Y. D., Antonellis, K. J., Scherf, U., et al. (2003) Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics.* **4**: 249–264.
4. Smyth, G. (2004) Linear models and empirical bayes methods for assessing differential expression in microarray experiments. *Stat Appl Genet Mol Biol.* **3**: Article3.
5. Benjamini, Y., and Hochberg, Y. (1995) Controlling false discovery rate: a practical and powerful approach to multiple testing. *J R Stat Soc Series B.* **57**: 289–300.
6. Nettleton, D. (2006) A discussion of statistical methods for design and analysis of microarray experiments for plant scientists. *Plant Cell.* **18**: 2112–2121.
7. Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., et al. (2000) Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nat Genet.* **25**: 25–29.
8. Clarke, J. D., and Zhu, T. (2006) Microarray analysis of the transcriptome as a stepping stone towards understanding biological systems: practical considerations and perspectives. *Plant J.* **45**: 630–650.
9. Allison, D. B., Cui, X., Page, G. P., and Sabripour, M. (2006) Microarray data analysis: from disarray to consolidation and consensus. *Nat Rev Genet.* **7**: 55–65.
10. Cordero, F., Botta, M., and Calogero, R. A. (2007) Microarray data analysis and mining approaches. *Brief Funct Genomic Proteomic.* **6**: 265–281.
11. Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., et al. (2004) Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol.* **5**: R80.
12. R Development Core Team. (2008) R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0. URL http://www.R-project.org.

13. Kilian, J., Whitehead, D., Horak, J., Wanke, D., Weinl, S., Batistic, O., et al. (2007) The AtGenExpress global stress expression data set: protocols, evaluation and model data analysis of UV-B light, drought and cold stress responses. *Plant J.* **50**: 347–363.
14. Falcon, S., and Gentleman, R. (2007) Using GOstats to test gene lists for GO term association. *Bioinformatics.* **23**: 257–258.
15. Swarbreck, D., Wilks, C., Lamesch, P., Berardini, T. Z., Garcia-Hernandez, M., Foerster, H., et al. (2008) The Arabidopsis Information Resource (TAIR): gene structure and function annotation. *Nucleic Acids Res.* **36**: D1009–D1014.
16. Wilson, C. L., and Miller, C. J. (2005) Simpleaffy: a BioConductor package for Affymetrix quality control and data analysis. *Bioinformatics.* **21**: 3683–3685.
17. Gautier, L., Cope, L., Bolstad, B. M., and Irizarry, R. A. (2004) affy – analysis of Affymetrix GeneChip data at the probe level. *Bioinformatics.* **20**: 307–315.
18. Wu, Z., Irizarry, R. A., Gentleman, R., Murillo, F. M., and Spencer, F. (2004) A model based background adjustment for oligonucleotide expression arrays. *J Am Stat Assoc.* **99**: 909–917.
19. Iliev, E. A., Xu, W., Polisensky, D. H., Oh, M. H., Torisky, R. S., Clouse, S. D., et al. (2002) Transcriptional and posttranscriptional regulation of Arabidopsis TCH4 expression by diverse stimuli. Roles of cis regions and brassinosteroids. *Plant Physiol.* **130**: 770–783.