Using R for the Analysis of DNA Microarray Data

by Sandrine Dudoit, Yee Hwa Yang, and Ben Bolstad

Overview

Microarray experiments generate large and complex multivariate datasets. Careful statistical design and analysis are essential to improve the efficiency and reliability of microarray experiments, from the early design and pre-processing stages to higher-level analyses. Access to an efficient, portable, and distributed statistical computing environment is a related and equally critical aspect of the analysis of gene expression data. As part of the Bioconductor project, we are working on the development of R packages implementing statistical methods for the design and analysis of DNA microarray experiments. An early effort is found in the **sma** (Statistics for Microarray Analysis) package which we wrote in the Fall of 2000. This small package was initially built to allow the sharing of software with our collaborators, both statisticians and biologists, for pre-processing tasks such as normalization and diagnostic plots. We are in the process of expanding the scope of this package by implementing new methods for microarray experimental design, normalization, estimation, multiple testing, cluster analysis, and discriminant analysis.

This short article begins with a brief introduction to the biology and technology of DNA microarray experiments. Next, we illustrate some of the main steps in the analysis of microarray gene expression data, using the apo AI experiment as a case study. Additional details can be found in Dudoit et al. (2002b). Although we focus primarily on two-color cDNA microarrays, a number of the proposed methods and implementations extend to other platforms as well (e.g., Affymetrix oligonucleotide chips, nylon membrane arrays). Finally, we discuss ongoing revisions and extensions to **sma** as part of the Bioconductor project. This project aims more generally to produce an open source and open development computing environment for biologists and statisticians working on the analysis of genomic data. More information on Bioconductor can be found at `http://www.bioconductor.org`.

Background on DNA microarrays

The ever-increasing rate at which genomes are being sequenced has opened a new area of genome research, *functional genomics*, which is concerned with assigning biological function to DNA sequences. With the availability of the DNA sequences of many genomes (e.g., the yeast *Saccharomyces cerevisae*, the round worm *Caenorhabditis elegans*, the fruit fly *Drosophila melanogaster*, and numerous bacteria) and the recent release of the first draft of the human genome, an essential and formidable task is to define the role of each gene and elucidate interactions between sets of genes. Innovative approaches, such as the cDNA and oligonucleotide microarray technologies, have been developed to exploit DNA sequence data and yield information about gene expression levels for entire genomes.

Different aspects of gene expression can be studied using microarrays, such as expression at the transcription or translation level, subcellular localization of gene products, and protein binding sites on DNA. To date, attention has focused primarily on expression at the transcription stage, i.e., on mRNA or transcript levels. There are several types of microarray systems, including the two-color cDNA microarrays developed in the Brown and Botstein labs at Stanford and the high-density oligonucleotide chips from the Affymetrix company; the brief description below focuses on the former.

cDNA microarrays consist of thousands of individual DNA sequences printed in a high-density array on a glass microscope slide using a robotic printer or arrayer. The relative abundance of these spotted DNA sequences in two DNA or RNA samples may be assessed by monitoring the *differential hybridization* of the two samples to the sequences on the array. For mRNA samples, the two samples or *targets* are reverse-transcribed into cDNA, labeled using different fluorescent dyes (usually a red-fluorescent dye, Cyanine 5 or Cy5, and a green-fluorescent dye, Cyanine 3 or Cy3), then mixed in equal proportions and hybridized with the arrayed DNA sequences or *probes* (following the definition of probe and target adopted in "The Chipping Forecast", a January 1999 supplement to Nature Genetics). After this competitive hybridization, the slides are imaged using a scanner and fluorescence measurements are made separately for each dye at each spot on the array. The ratio of the red and green fluorescence intensities for each spot is indicative of the relative abundance of the corresponding DNA probe in the two nucleic acid target samples. See Brown and Botstein (1999) for a more detailed introduction to the biology and technology of cDNA microarrays.

DNA microarray experiments raise numerous statistical questions, in fields as diverse as experimental design, image analysis, multiple testing, cluster analysis, and discriminant analysis. The main

steps in the statistical design and analysis of a microarray experiment are summarized in Figure 1. Each step in this process depends critically on the availability of an efficient, portable, and distributed statistical computing environment.

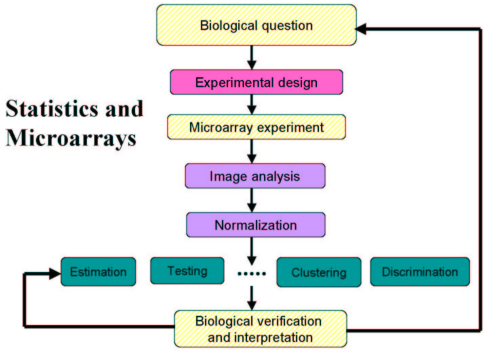

Figure 1: Main steps in the statistical design and analysis of a DNA microarray experiment (solid boxes).

Sample microarray dataset

We will illustrate some of the main steps in the analysis of a microarray experiment using gene expression data from the apo AI experiment described in detail in Callow et al. (2000). Apolipoprotein AI (apo AI) is a gene known to play a pivotal role in HDL metabolism, and mice with the apo AI gene knocked-out have very low HDL cholesterol levels. The goal of this experiment was to identify genes with altered expression in the livers of apo AI knock-out mice compared to inbred control mice. The treatment group consisted of eight mice with the apo AI gene knocked-out and the control group consisted of eight control C57Bl/6 mice. For each of these sixteen mice, target cDNA was obtained from mRNA by reverse transcription and labeled using a red-fluorescent dye, Cy5. The reference sample used in all hybridizations was prepared by pooling cDNA from the eight control mice and was labeled with a green-fluorescent dye, Cy3. Target cDNA was hybridized to microarrays containing 6,384 DNA probes, including 257 related to lipid metabolism. Microarrays were printed using 4×4 print-tips and are thus partitioned into a 4×4 grid matrix (the terms grid, sector, and print-tip group are used interchangeably in the microarray literature). Each grid consists of a 19×21 spot matrix that was printed with a single print-tip.

Raw images and background corrected Cy3 and Cy5 fluorescence intensities for all sixteen hybridizations are available at `http://www.stat.berkeley.edu/users/terry/zarray/Html/apodata.html`. Fluorescence intensity data from processed images for six of the sixteen hybridizations (three knock-out mice and three control mice) were also included in the **sma** package and can be accessed with the command `data(MouseArray)`. Figure 2 displays an RGB overlay of the Cy3 and Cy5 images for one of the sixteen arrays (files 1230ko1G.tif.marray and 1230ko1R.tif.marray corresponding to knock-out mouse 1, i.e., `mouse4` array in the dataset `MouseArray`). Analyses described below were performed using **sma** version 0.5.7 and R version 1.3.1.

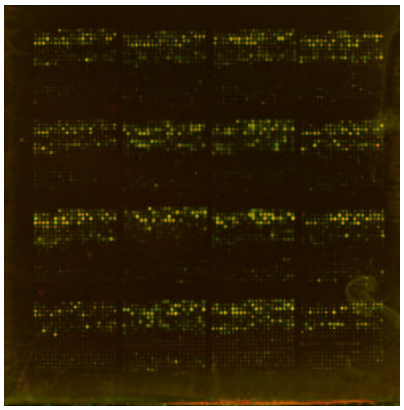

Figure 2: RGB overlay of the Cy3 and Cy5 images for knock-out mouse 1, i.e., `mouse4` array in the dataset `MouseArray`.

Pre-processing

Image analysis

The *raw* data from a microarray experiment are the image files produced by the scanner; these are typically pairs of 16-bit tagged image file format (TIFF) files, one for each fluorescent dye (the images for the apo AI experiment are about 2MB in size; more recent images produced from higher resolution scans are around 10 to 20 MB). Image analysis is required to extract foreground and background fluorescence intensity measurements for each spotted DNA sequence. Widely used commercial image processing software packages include **GenePix** for the Axon scanner and **ImaGene** from BioDiscovery; freely available software includes **ScanAlyze**. Image processing is beyond the scope of this article, and the reader is referred to Yang et al. (2002a) for a detailed discussion of microarray image analysis and additional references.

For the apo AI experiment, each of the sixteen hybridizations produced a pair of 16-bit images. These images were processed using the software package **Spot** (Buckley, 2000). **Spot** is built on R and is a specialized version of another R package called **VOIR**, which is being developed by the CSIRO Image Analysis Group (`http://www.cmis.csiro.au/iap/spot.htm`). The segmentation component, based on a seeded region growing algorithm, places no restriction on the size or shape of the spots. The background adjustment method relies on morphological

opening to generate an image of the estimated background intensity for the entire slide. The results of an analysis by **Spot** of microarray images are returned as a data frame and can thus immediately be displayed, manipulated, and analyzed in a number of ways within R. The rows of this data frame correspond to the spots on the array and the columns to different spot statistics: e.g., grid row and column coordinates, spot row and column coordinates, red and green foreground intensities, red and green background intensities for different background adjustment methods, spot area, perimeter. The six data frames `mouse1`, ..., `mouse6`, in `MouseArray` contain **Spot** output for three control mice and three knockout mice, respectively. In what follows, we use R and G to denote the background corrected red and green fluorescence intensities of a particular spot, and M to denote the corresponding base-2 log-ratio, $\log_2 R/G$.

Reading in data

In general, microarray image processing results are stored in ASCII files and can be loaded into R using specialized functions like `read.spot` and `read.genepix` for **Spot** and **GenePix** output, respectively. These functions are simply *wrapper* functions around `read.table`, which take into account the specific file format for output from different image processing software packages.

We have written a number of initialization functions to manipulate the image analysis output for batches of arrays, i.e., collections of slides with the same spot layout. The function `init.grid` is an interactive function for specifying the dimensions of the spot and grid matrices. These parameters are determined by the printing layout of the array, and are used for the spatial representation of spot statistics in the function `plot.spatial` and the within-print-tip-group normalization procedure implemented in `stat.ma`. The function `init.data` is an interactive function which creates a data structure for multi-slide microarray experiments. The data structure is a list of four matrices, storing raw red and green foreground intensities, and red and green background intensities. The rows of the matrices correspond to spotted DNA sequences and the columns to hybridizations (i.e., arrays or slides). The function also allows the user to add hybridization data to an existing list. The following commands may be used to extract red and green fluorescence intensities from the image analysis output for the six arrays in `MouseArray`, and compute intensity log-ratios and average log-intensities.

```
data(MouseArray)
mouse.setup <- init.grid()
mouse.data <- init.data()
mouse.MA0 <- stat.ma(mouse.data, mouse.setup,
                     norm="n")
```

Eventually, we would like to retrieve the image analysis results directly from a laboratory database. In addition, to allow more general and systematic representation and manipulation of microarray data, we are working on the definition of new classes of R objects for microarrays using the class/method mechanism in the **methods** package (cf. section on ongoing projects).

Diagnostic plots

Before proceeding to normalization or any higher-level analysis, it is instructive to look at diagnostic plots of spot statistics, such as red and green foreground and background log-intensities, intensity log-ratio, area, etc.

Spatial plots of spot statistics. The **sma** function `plot.spatial` creates an image of shades of gray or colors that represents the values of a statistic for each spot on the array. This function can be used to explore whether there are any spatial effects in the data, for example, print-tip or cover-slip effects. The commands

```
M <- stat.ma(mouse.data, mouse.setup,
             norm="n")$M[,4]
plot.spatial(M, mouse.setup, crit1=1)
```

produce an image of the pre-normalization log-ratios M for the array `mouse4`. To display only the spots having the highest and lowest 5% pre-normalization log-ratios M, set the argument `crit1` to 0.05. Figure 3 clearly indicates that the lowest row of grids has high red intensities compared to green, and thus suggests the existence of spatially-dependent dye biases.

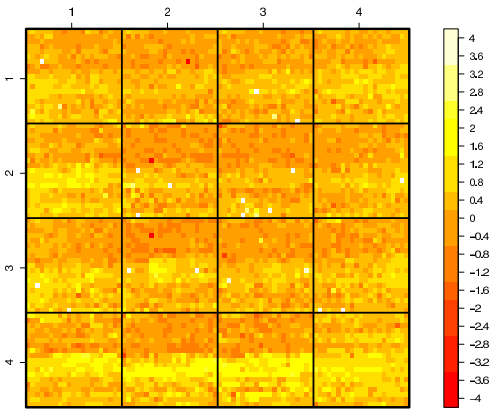

Figure 3: Image of the pre-normalization intensity log-ratios for the `mouse4` array using the `heat.colors` palette (`plot.spatial` output).

Boxplots. Boxplots of spot statistics by plate, sector, or slide can also be useful to identify spot or hybridization artifacts. The boxplots in Figure 4 again

suggest the existence of spatially-dependent dye biases.

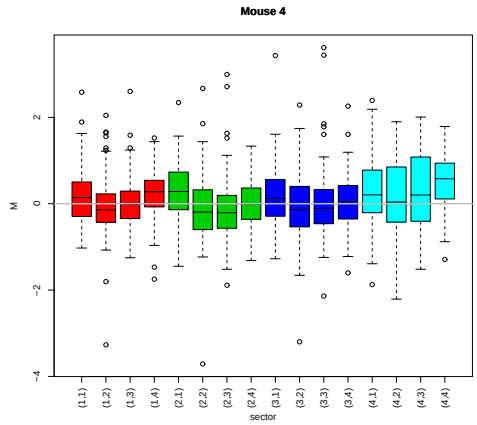

Figure 4: Boxplots by sector of the pre-normalization intensity log-ratios for the `mouse4` array. The pairs (i, j), $i,\ j = 1, \ldots, 4$, refer to sector or grid coordinates in the 4×4 grid matrix shown in Figure 2.

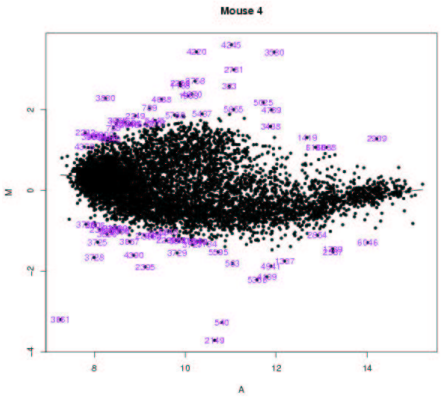

Figure 5: Pre-normalization MA-plot for the `mouse4` array, highlighting spots with the highest and lowest 0.5% log-ratios (`plot.mva` output).

MA-plots. Single-slide expression data are typically displayed by plotting the log-intensity $\log_2 R$ in the red channel vs. the log-intensity $\log_2 G$ in the green channel. Such plots tend to give an unrealistic sense of concordance between the red and green intensities and can mask interesting features of the data. We thus prefer to plot the intensity log-ratio $M = \log_2 R/G$ vs. the mean log intensity $A = \log_2 \sqrt{RG}$. An MA-plot amounts to a 45^o counterclockwise rotation of the $(\log_2 G, \log_2 R)$-coordinate system, followed by scaling of the coordinates. It is thus another representation of the (R, G) data in terms of the log-ratios M which directly measure differences between the red and green channels and are the quantities of interest to most investigators. We have found MA-plots to be more revealing than their $\log_2 R$ vs. $\log_2 G$ counterparts in terms of identifying spot artifacts and for normalization purposes (Dudoit et al., 2002b; Yang et al., 2001, 2002b). For the apo AI experiment,

```
plot.mva(mouse.data,mouse.setup,norm="l",
   image.id=4,extra.type="tci",plot.type="r",
   crit1=0.005,col.ex="purple",pch=20)
title("Mouse 4")
```

produces an MA-plot for the `mouse4` array, highlighting spots with the highest and lowest 0.5% log-ratios using the corresponding row number in `mouse.data` (Figure 5).

Normalization

The purpose of normalization is to identify and remove sources of systematic variation, other than differential expression, in the measured fluorescence intensities (e.g., different labeling efficiencies and scanning properties of the Cy3 and Cy5 dyes; different scanning parameters, such as PMT settings; print-tip, spatial, or plate effects). It is necessary to normalize the fluorescence intensities before any analysis which involves comparing expression levels within or between slides (e.g., clustering, multiple testing). The need for normalization can be seen most clearly in self-self experiments, in which two identical mRNA samples are labeled with different dyes and hybridized to the same slide (Dudoit et al., 2002b). Although there is no differential expression and one expects the red and green intensities to be equal, the red intensities often tend to be lower than the green intensities. Furthermore, the imbalance in the red and green intensities is usually not constant across the spots within and between arrays, and can vary according to overall spot intensity, location on the array, plate origin, and possibly other variables. Figure 6 displays the pre-normalization MA-plot for the `mouse4` array, with the sixteen lowess fits for each of the print-tip-groups (using a smoother span $f = 0.3$ for the `lowess` function). The plot was produced using the function `plot.print.tip.lowess`

```
col <- sort(rep(2:5,4))
lty <- rep(1:4,4)
labs <- paste("(",sort(rep(1:4,4)),",",
   rep(1:4,4),")",sep="")
plot.print.tip.lowess(mouse.data,mouse.setup,
   norm="n",image.id=4,pch=20,lwd=3,
   palette=col,lty.palette=lty,cex=0.6)
legend(10.5,-2.5,legend=labs[order(lty,col)],
   col=col[order(lty,col)],
   lty=lty[order(lty,col)], ncol=4,lwd=2)
title("Mouse 4")
```

Figure 6 illustrates the non-linear dependence of the log-ratio M on the overall spot intensity A and thus suggests that an intensity or A-dependent normalization method is preferable to a global one (e.g., median normalization). Also, four lowess curves clearly stand out from the remaining twelve curves, suggesting strong print-tip or spatial effects. The four curves correspond to the last row of grids in the 4×4 grid matrix.

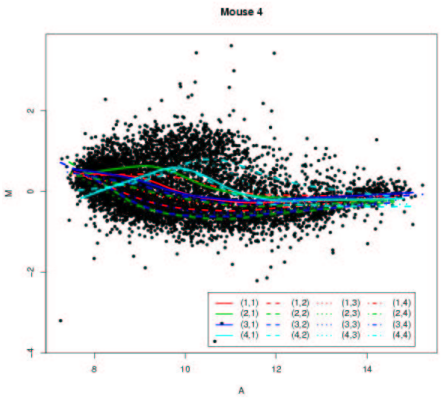

Figure 6: Pre-normalization MA-plot for the `mouse4` array, with the lowess fits for individual print-tips or sectors. Different colors are used to represent lowess curves for print-tips from different rows, and different line types are used to represent lowess curves for print-tips from different columns (`plot.print.tip.lowess` output).

We have developed location and scale normalization methods which correct for intensity and spatial dye biases using *robust local regression* (Yang et al., 2001, 2002b). These procedures are implemented in the **sma** function `stat.ma`, which uses the R function `lowess` to perform robust local regression of the log-ratio M on spot intensity A. Five methods are currently available for normalizing the log-ratios; the method is specified using the `norm` argument of the function `stat.ma`. These five options are: `"n"` for no normalization between the two channels; `"m"` for global median normalization, which sets the median of the intensity log-ratios M to zero; `"l"` for global lowess normalization, i.e., regression of M on A for all spots; `"p"` for within-print-tip-group lowess normalization; and `"s"` for scaled within-print-tip-group normalization. For the `"s"` option, scaling is done by the median absolute deviation (MAD) of the lowess location normalized log-ratios for each print-tip-group. The code

```
mouse.lratio <-
  stat.ma(mouse.data, mouse.setup, norm="p")
```

performs within-print-tip-group lowess location normalization for the six slides in `MouseArray`, and returns normalized log-ratios M and average log-intensities A. For the apo AI experiment, only a small proportion of the spots were expected to vary in intensity between the two channels; normalization was thus performed using all 6,384 probes. The function returns a list with two components: `M`, a matrix of normalized log-ratios, and `A`, a matrix of average log-intensities. The rows of these matrices correspond to spotted DNA sequences and the columns to the different slides.

Image analysis and normalization are the two main pre-processing steps required in any microarray experiment. For our purpose, normalized microarray data consist primarily of pairs (M, A) of intensity log-ratios and average log-intensities for each spot in each of several slides. In addition to these basic statistics, we are planning on deriving spot and slide quality statistics and incorporating these in further analyses (cf. section on ongoing projects).

Main statistical analysis

The phrase "main statistical analysis" refers to the application of statistical methodology to normalized intensity data in order to answer the biological question for which the microarray experiment was designed. Appropriate statistical methods depend largely on the question of interest to the investigator and in general can span the entire field of Statistics. For model organisms such as mice or yeast, biological questions of interest might include the identification of differentially expressed genes in factorial experiments studying the simultaneous gene expression response to factors such as drug treatment, cell type, spatial location of tissues, and time. In human cancer microarray studies, one may be interested in the discovery of new tumor subclasses, or in the identification of genes that are good predictors of clinical outcomes such as survival or response to treatment. There is clearly no general "next step" in the analysis of microarray data, and the implementation of statistical methodology will require the development of question specific R packages (cf. section on ongoing projects). Next, we discuss differential expression, an important and common question in microarray experiments.

Differential expression

Differentially expressed genes are genes whose expression levels are associated with a response or covariate of interest. In general, the covariates could be either polytomous (e.g., treatment/control status, cell type, drug type) or continuous (e.g., dose of a drug, time). Similarly, the responses could be either polytomous or continuous, for example, tumor type, response to treatment, or censored survival time in clinical applications of microarrays. An approach to the identification of differentially expressed genes is to compute, for each gene, a statistic assessing the strength of the association between the expression levels and responses or covariates. In such one gene at a time approaches, genes are ranked based on these statistics and a subset is typically selected for further biological verification. This gene subset may be chosen based on either the number of genes that can be conveniently followed-up (and subject matter knowledge), or statistical significance as described in the next section. The package **sma** contains a number of functions for computing and plotting frequentist and Bayesian statistics for each gene in a microarray

experiment.

Single-slide methods. Single-slide experiments aim to compare transcript abundance in two mRNA samples, the red and green labeled mRNA samples hybridized to the same slide. The **sma** functions `stat.Chen`, `stat.Newton`, and `stat.ChurSap` implement, respectively, the single-slide methods of Chen et al. (1997), Newton et al. (2001), and Sapir and Churchill (2000). These methods are based on assumed parametric models for the (R, G) intensity pairs (e.g., Gamma or Gaussian models). The resulting model-dependent rules amount to drawing two curves in the (R, G)-plane and calling a gene differentially expressed if its (R, G) measured intensities fall outside the region between the two curves. The above three functions can be applied individually, with options for producing *MA*-plots showing the model cut-offs. Alternatively, the three methods can be compared graphically as in Figure 7, by calling `plot.single.slide(mouse.data, mouse.setup, norm="p", image.id=4)`. This produces an *MA*-plot, print-tip-group lowess location normalized, for the `mouse4` array with cut-offs from each of the three single-slide methods. Black curves correspond to 1:1, 1:10, 1:100 log-odds ratios for the Newton et al. method; dashed lines are the 95% and 99% confidence limits from the Chen et al. method; dotted lines are the 95% and 99% posterior probability cut-offs from the Sapir and Churchill method.

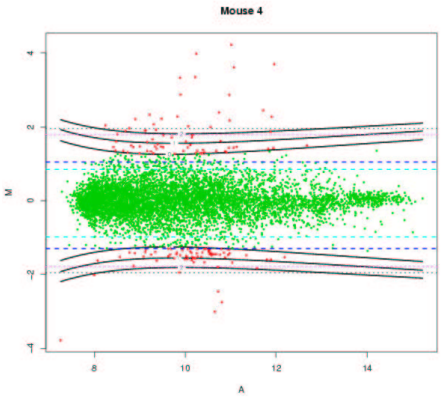

Figure 7: *MA*-plot with cut-offs for three single-slide methods applied to the `mouse4` array (`plot.single.slide` output).

Multiple-slide methods. Multiple-slide experiments typically involve the comparison of transcript abundance in two or more types of mRNA samples hybridized to different slides. **sma** functions for multiple-slide analyses include: `stat.bayesian`, which computes odds of differential expression under a Bayesian model for gene expression; `stat.t2`, which computes two-sample *t*-statistics for comparing the expression response in two groups of mRNA samples. The multiple testing package **multtest** contains additional functions for computing test statistics (paired *t*-statistics, *F*-statistics, etc.) and associated permutation adjusted *p*-values for each gene on the array (see next section). For the apo AI data, the following commands implement two possible approaches for comparing expression levels in the livers of knock-out and control mice.

```
## Two-sample Welch t-statistics
## Comparison of the expression levels of the
## three control and three knock-out mice
## (data from mouse1, ..., mouse6)
cl <- c(rep(1,3), rep(2,3))
mouse.t2 <- stat.t2(mouse.lratio, cl)
plot.t2(mouse.t2, "Apo AI experiment, six mice")

## Bayesian odds ratio
## Comparison of the expression levels of the
## three knock-out mice to the polled controls
## (data from mouse4, mouse5, mouse6)
mouse.bayesian <-
  stat.bayesian(M=mouse.lratio$M[,4:6])
plot(mouse.bayesian$Xprep$Mbar,
     mouse.bayesian$lods)
```

Multiple testing

The biological question of differential expression can be restated as a problem in *multiple hypothesis testing*: the simultaneous test for each gene of the null hypothesis of no association between the expression levels and the responses or covariates. As a typical microarray experiment measures expression levels for thousands of genes simultaneously, we are faced with an extreme form of multiple testing when assessing the statistical significance of the results.

The **multtest** package implements a number of resampling-based multiple testing procedures. It includes procedures for controlling the family-wise Type I error rate (FWER): Bonferroni, Hochberg, Holm, Šidák, Westfall and Young minP and maxT procedures. The Westfall and Young procedures take into account the joint distribution of the test statistics and are thus in general less conservative than the other four procedures. The **multtest** package also includes procedures for controlling the false discovery rate (FDR): Benjamini and Hochberg and Benjamini and Yekutieli step-up procedures (see Dudoit et al. (2002a) for a review of multiple testing procedures and complete references). These procedures are implemented for tests based on *t*-statistics, *F*-statistics, paired *t*-statistics, block *F*-statistics, Wilcoxon statistics. The results of the procedures are summarized using *adjusted p-values*, which reflect for each gene the overall experiment Type I error rate when genes with a smaller *p*-value are declared differentially expressed. The *p*-values may be obtained either from the nominal distribution of the test statistics or by permutation.

For the apo AI experiment, differentially expressed genes between the eight knock-out and eight control mice were identified by computing two-sample Welch's t-statistics for each gene. In order to assess the statistical significance of the results, four multiple testing procedures (Holm, Westfall and Young maxT, Benjamini and Hochberg FDR, Benjamini and Yekutieli FDR) were considered, and unadjusted and adjusted p-values were estimated based on all possible $\binom{16}{8} = 12,870$ permutations of the knock-out/control labels. Figure 8 displays plots of the ordered adjusted p-values for these four multiple testing procedures. The functions `mt.maxT`, `mt.rawp2adjp`, and `mt.plot` from the **multtest** package were used to compute and display adjusted p-values.

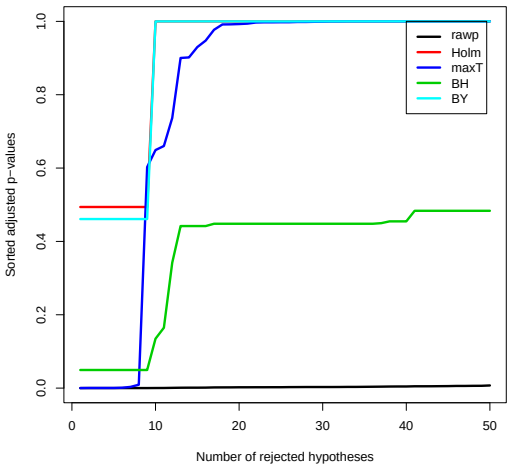

Figure 8: Plot of sorted adjusted p-values for the apo AI experiment (**multtest** package, `mt.plot` output).

Biological verification

In the apo AI experiment, eight spotted DNA sequences clearly stood out from the remaining sequences and had maxT adjusted p-values less than 0.01. These eight sequences correspond to only four distinct genes: apo AI (3 copies), apo CIII (2 copies), sterol C5 desaturase (2 copies), and a novel EST (1 copy). All changes were confirmed by real-time quantitative PCR (RT-PCR) as described in Callow et al. (2000). The presence of apo AI among the differentially expressed genes is to be expected, as this is the gene that was knocked out in the treatment mice. The apo CIII gene, also associated with lipoprotein metabolism, is located very close to the apo AI locus; Callow et al. (2000) showed that the down-regulation of apo CIII was actually due to genetic polymorphism rather than lack of apo AI. The presence of apo AI and apo CIII among the differentially expressed genes thus provides a check of the statistical method, if not a biologically interesting finding. Sterol C5 desaturase is an enzyme which catalyzes one of the terminal steps in cholesterol synthesis and the novel EST shares sequence similarity to a family of ATPases. Note that the application of single-slide methods to array data from individual knock-out mice generally produced a large number of false positives, while at the same time missing some of the confirmed genes.

Ongoing projects

The **sma** package represents a preliminary and limited effort toward the goal of providing a statistical computing environment for the analysis of microarray data. Modifications and extensions to **sma** will be done within the Bioconductor project (`http://www.bioconductor.org`). The Bioconductor project aims more generally to produce an open source and open development computing environment for biologists and statisticians working on the analysis of genomic data. Important changes include the use of the class/method mechanism in the R **methods** package to allow more general and systematic representation and manipulation of microarray data (Bioconductor **Biobase** package). In addition, we are in the process of splitting and expanding the **sma** package into the following task specific packages.

Experimental design (**MarrayDesign**)

This package will include functions for: setting up a design matrix for a given experimental design (this design matrix can be used to efficiently combine data across slides with functions similar to `lm` or `glm`); searching for optimal designs; and creating graphical representations of the design choices.

Spot quality (**MarrayQual**)

Building upon our previous work on image analysis and normalization, we plan to devise spot and slide quality statistics and incorporate these statistics in further analyses of the fluorescence intensity data, by weighting the contribution of each spot based on its quality.

Normalization (**MarrayNorm**)

This package contains functions for diagnostic plots and normalization procedures based on robust local regression. Normalization procedures were extended to accommodate a broader class of dye biases and the use of control sequences spotted on the array and possibly spiked in the target cDNA samples.

Cluster analysis (**MarrayClust**)

R already has an extensive collection of clustering procedures in packages such as **cluster**, **mva**, and **GeneSOM**. Building upon these existing packages, we are implementing new methods which are useful for the clustering

of microarray data, genes or mRNA samples. These include resampling based-methods for estimating the number of clusters, for improving the accuracy of a clustering procedure, and for assessing the accuracy of cluster assignments for individual observations (Fridlyand and Dudoit, 2001).

Class prediction (**MarrayPred**)

This package builds upon existing R packages such as **class** and **rpart** and contains functions for bagging and boosting classifiers. The functions also return estimates of misclassification rates and measures assessing the confidence of predictions for individual observations. Resampling-based procedures for variable selection as described in Breiman (1999) will also be implemented in the package.

Affymetrix chips (**affy**)

Low-level analysis of Affymetrix data, such as normalization and calculation of expression estimates, is being handled by the **affy** package `http://biosun01.biostat.jhsph.edu/~ririzarr/Raffy/index.html`.

The packages **MarrayNorm**, **affy**, and **MarrayPred** are closest to being released. Required developments to increase the effectiveness of these R packages within a biological context include the following.

Links to biological databases: An important aspect of the analysis of microarray experiments is the seamless access to biological information resources such as the National Center for Biotechnology Information (NCBI) Entrez system (`http://www.ncbi.nlm.nih.gov/Entrez/`) or the Gene Ontology (GO) Consortium (`http://www.geneontology.org`). The Bioconductor **annotate** package developed by Robert Gentleman already provides browser access to LocusLink and GenBank.

GUI: Not all biologist users will feel comfortable with the command line R environment. To accommodate a broad class of users and allow easy access to the statistical methodology, the computing environment should provide a graphical user interface.

Documentation: We envision two main classes of users for these packages: biologists performing microarray experiments, and researchers involved in the development of statistical methodology for microarray experiments. The former, will be primarily users of a standard set of functions from these packages. They will need guidance in deciding which statistical approaches and packages may be appropriate for their experiments, in choosing among the various options provided by the functions, and in correctly interpreting the results of their visualizations and statistical analyses. Extended tutorials, with step-by-step analyses of microarray experiments, will be provided. The **Sweave** package to be included in R 1.5 will be used for automatic report generation mixing text and R code. Researchers in the second group, will likely be interested in writing their own functions and packages, in addition to using existing functions. The Bioconductor project will provide a framework for integrating their efforts.

Thanks to Natalie Thorne, Ingrid Lönnstedt, and Jessica Mar for their contributions to the **sma** package.

Bibliography

L. Breiman. Random forests - random features. Technical Report 567, Department of Statistics, U.C. Berkeley, 1999. 31

P. O. Brown and D. Botstein. Exploring the new world of the genome with DNA microarrays. In *The Chipping Forecast*, volume 21, pages 33–37. Supplement to Nature Genetics, January 1999. 24

M. J. Buckley. *The Spot user's guide*. CSIRO Mathematical and Information Sciences, August 2000. `http://www.cmis.csiro.au/IAP/spot.htm`. 25

M. J. Callow, S. Dudoit, E. L. Gong, T. P. Speed, and E. M. Rubin. Microarray expression profiling identifies genes with altered expression in HDL deficient mice. *Genome Research*, 10(12):2022–2029, 2000. 25, 30

Y. Chen, E. R. Dougherty, and M. L. Bittner. Ratio-based decisions and the quantitative analysis of cDNA microarray images. *Journal of Biomedical Optics*, 2:364–374, 1997. 29

S. Dudoit, J. P. Shaffer, and J. C. Boldrick. Multiple hypothesis testing in microarray experiments. Submitted, 2002a. 29

S. Dudoit, Y. H. Yang, M. J. Callow, and T. P. Speed. Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. *Statistica Sinica*, 12(1), 2002b. 24, 27

J. Fridlyand and S. Dudoit. Applications of resampling methods to estimate the number of clusters and to improve the accuracy of a clustering method. Technical Report 600, Department of Statistics, U.C. Berkeley, September 2001. 31

M. A. Newton, C. M. Kendziorski, C. S. Richmond, F. R . Blattner, and K. W. Tsui. On differential variability of expression ratios: Improving statistical inference about gene expression changes from microarray data. *Journal of Computational Biology*, 8: 37–52, 2001. 29

M. Sapir and G. A. Churchill. *Estimating the posterior probability of differential gene expression from microarray data*. Poster, The Jackson Laboratory, 2000. `http://www.jax.org/research/churchill/`. 29

Y. H. Yang, M. J. Buckley, S. Dudoit, and T. P. Speed. Comparison of methods for image analysis on cDNA microarray data. *Journal of Computational and Graphical Statistic*, 11(1), 2002a. 25

Y. H. Yang, S. Dudoit, P. Luu, D. M. Lin, V. Peng, J. Ngai, and T. P. Speed. Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Research*, 30(4), 2002b. 27, 28

Y. H. Yang, S. Dudoit, P. Luu, and T. P. Speed. Normalization for cDNA microarray data. In Michael L. Bittner, Yidong Chen, Andreas N. Dorsel, and Edward R. Dougherty, editors, *Microarrays: Optical Technologies and Informatics*, volume 4266 of *Proceedings of SPIE*, May 2001. 27, 28

Sandrine Dudoit
University of California, Berkeley
http://www.stat.berkeley.edu/~sandrine
sandrine@stat.berkeley.edu

Yee Hwa (Jean) Yang
University of California, Berkeley
yeehwa@stat.berkeley.edu

Ben Bolstad
University of California, Berkeley
bolstad@stat.berkeley.edu

Changes in R 1.4.0

by the R Core Team

The 'NEWS' file of the R sources and hence this article have been reorganized into several sections (user-visible changes, new features, deprecated & defunct, documentation changes, utilities, C-level facilities, bug fixes) in order to provide a better overview of all the changes in R 1.4.0.

User-visible changes

This is a new section to highlight changes in behaviour, which may be given in more detail in the following sections. Many bug fixes are also user-visible changes.

- The default save format has been changed, so saved workspaces and objects cannot (by default) be read in earlier versions of R.
- The number of bins selected by default in a histogram uses the correct version of Sturges' formula and will usually be one larger.
- `data.frame()` no longer converts logical arguments to factors (following S4 rather than S3).
- `read.table()` has new arguments 'nrows' and 'colClasses'. If the latter is `NA` (the default), conversion is attempted to logical, integer, numeric or complex, not just to numeric.
- `model.matrix()` treats logical variables as a factors with levels `c(FALSE, TRUE)` (rather than 0-1 valued numerical variables). This makes R compatible with all S versions.
- Transparency is now supported on most graphics devices. This means that using `par("bg")`, for example in `legend()`, will by default give a transparent rather than opaque background.
- [dpqr]gamma now has third argument 'rate' for S-compatibility (and for compatibility with exponentials). Calls which use positional matching may need to be altered.
- The meaning of `spar = 0` in `smooth.spline()` has changed.
- `substring()` and `substring<-()` do nothing silently on a character vector of length 0, rather than generating an error. This is consistent with other functions and with S.
- For compatibility with S4, any arithmetic operation using a zero-length vector has a zero-length result. (This was already true for logical operations, which were compatible with S4 rather than S3.)
- `undoc()` and `codoc()` have been moved to the new package **tools**.
- The name of the site profile now defaults to 'R_HOME/etc/Rprofile.site'.