Lyng, H., Badiee, A., Svendsrud, D. H., Hovig, E., Myklebost, O., and Stokke, T. (2004). Profound influence of microarray scanner characteristics on gene expression ratios: Analysis and procedure for correction. *BMC Genom.* **5,** 10.

Parkinson, H., Sarkans, U., Shojatalab, M., Abeygunawardena, N., Contrino, S., Coulson, R., Farne, A., Garcia Lara, G., Holloway, E., Kapushesky, M., Lilja, P., Mukherjee, G., Oezcimen, A., Rayner, T., Rocca-Serra, P., Sharma, A., Sansone, S., and Brazma, A. (2005). ArrayExpress: A public repository for microarray gene expression data at the EBI. *Nucleic Acids Res.* **33,** D553–D555.

R Development Core Team (2005). R: A language and environment for statistical computing R Foundation for Statistical Computing, Vienna, Austria.

Rhee, S. J., Walker, W. A., and Cherayil, B. J. (2005). Developmentally regulated intestinal expression of IFN-gamma and its target genes and the age-specific response to enteric Salmonella infection. *J. Immunol.* **175,** 1127–1136.

Saal, L., Troein, C., Vallon-Christersson, J., Gruvberger, S., Borg, Å., and Peterson, C. (2002). BioArray Software Environment (BASE): A platform for comprehensive management and analysis of microarray data. *Genome Biol.* **3,** software0003.1-0003.6.

Sollier, J., Lin, W., Soustelle, C., Suhre, K., Nicolas, A., Geli, V., and de La Roche Saint-Andre, V. (2004). Set1 is required for meiotic S-phase onset, double-strand break formation and middle gene expression. *EMBO J.* **23,** 1957–1967.

Spellman, P. T., Miller, M., Stewart, J., Troup, C., Sarkans, U., Chervitz, S., Bernhart, D., Sherlock, G., Ball, C., Lepage, M., Swiatek, M., Marks, W. L., Goncalves, J., Markel, S., Iordan, D., Shojatalab, M., Pizarro, A., White, J., Hubley, R., Deutsch, E., Senger, M., Aronow, B. J., Robinson, A., Bassett, D., Stoeckert, C. J., Jr., and Brazma, A. (2002). Design and implementation of microarray gene expression markup language (MAGE-ML). *Genome Biol.* **3,** research0046.1-0046.9.

Sturme, M. H., Nakayama, J., Molenaar, D., Murakami, Y., Kunugi, R., Fujii, T., Vaughan, E. E., Kleerebezem, M., and de Vos, W. M. (2005). An agr-like two-component regulatory system in *Lactobacillus plantarum* is involved in production of a novel cyclic peptide and regulation of adherence. *J. Bacteriol.* **187,** 5224–5235.

Wang, X., Wang, M., Amarzguioui, M., Liu, F., Fodstad, O., and Prydz, H. (2004). Downregulation of tissue factor by RNA interference in human melanoma LOX-L cells reduces pulmonary metastasis in nude mice. *Int. J. Cancer* **112,** 994–1002.

# [8] Bioconductor: An Open Source Framework for Bioinformatics and Computational Biology

*By* MARK REIMERS and VINCENT J. CAREY

## Abstract

This chapter describes the Bioconductor project and details of its open source facilities for analysis of microarray and other high-throughput biological experiments. Particular attention is paid to concepts of container and workflow design, connections of biological metadata to statistical analysis products, support for statistical quality assessment, and calibration of inference uncertainty measures when tens of thousands of simultaneous statistical tests are performed.

METHODS IN ENZYMOLOGY, VOL. 411 0076-6879/06 $35.00

DOI: 10.1016/S0076-6879(06)11008-3

## Introduction: Bioconductor in Brief

Bioconductor is a project devoted to the development of software and methods for statistical analysis and visualization of data from high-throughput experimental platforms in biology. The project is fully open source, with most software released under the Lesser GNU Public License (see http://www.gnu.org/licenses/licenses.html#LGPL). Most software available through the Bioconductor project is written in R, an open source data analysis environment that has become a major tool for quantitative scientists throughout the world.

Bioconductor may be viewed as a collection of specifications of containers and workflows for preprocessing and analyzing high-throughput data.

- Containers are defined for management and analysis of expression data at various levels of technical processing, for management of experiment-level metadata in the Minimum Information about a Microarray Experiment (MIAME) data model (see later; Brazma *et al.*, 2006), for management and analysis of sample-level data and custom metadata about samples, and for the organization and manipulation of large quantities of biological annotation, such as mappings between proprietary probe identifiers and public database or ontology identifiers.
- Workflows are defined through high-level graphical user interfaces for specific forms of preprocessing and downstream analysis and through the interaction of software packages that are driven in a command-line interface. Documentation of workflows and workflow components is provided in the form of manual pages for specific modules and functions, vignettes that document multistep processes, and fully worked use case descriptions that are distributed with the software.
- Motivations and approaches of Bioconductor have been described comprehensively by Gentleman *et al.* (2004). This chapter provides details of philosophy, use, and future prospects of Bioconductor as a source of software and software design and distribution methods for critical methods of bioinformatics and computational biology.

## Technical Details

### *Software Distribution*

Bioconductor is rooted in the R language (Ihaka and Gentleman, 1996). Software resources are organized into packages, which are structured folders of code, documentation, and illustrative data. The distribution of packages for Bioconductor can proceed in various ways. At present, Windows and Macintosh graphical user interfaces for R include buttons

for package installation over the web. These buttons allow selection of the Bioconductor software repository and selection of specific packages. Alternatively, a user may load a publicly available script (http://www.bioconductor.org/biocLite.R) into an R session, issue the command `biocLite( )`, and software will travel over the internet into this R distribution, where it is installed automatically and will persist until removed manually. Software modules can also be downloaded manually using a web browser pointed at http://www.bioconductor.org or can be obtained using functions provided in a Bioconductor package called *reposTools*.

*Containers*

There are four basic container types at present. Containers are available for preprocessed microarray data (e.g., data imported from scanner outputs or Affymetrix CEL files), for postprocessed microarray data (including detailed information on sample characteristics and treatments), for metadata about microarray experiments (principally satisfying the MIAME protocol), and for general biological metadata, such as the Gene Ontology.

The use of containers depends on the internal structures of the containers, defined in Bioconductor infrastructure packages such as Biobase, and on the accessor methods that are provided in these infrastructure packages. An accessor `F` for a container `C` is used with the syntax `F(C)`.

*Preprocessed Microarray Data.* Affymetrix distributes a collection of CEL files from a Latin square design spike-in experiment. A subset of these data is distributed in the *SpikeInSubset* package of Bioconductor. The following code loads this package, loads the U133A-TAG subset provided there, and requests a report on this subset.

```
> library(SpikeInSubset)
> data(spikein133)
> spikein133

AffyBatch object
size of arrays=712x712 features (23781 kb)
cdf=HG-U133A_tag (22300 affyids)
number of samples=6
number of genes=22300
annotation=hgu133atag
```

This structure includes metadata about the samples. These metadata, often referred to as ''phenotype data,'' even though they can involve information not typically regarded as phenotypic, can be accessed in the form of an R data.frame using the pData accessor. Here we inquire about the dimensions of

the pData component and select, using the matrix [row, column] selection idiom, a small fraction of the metadata available for the spike-in study. This is organized as samples in rows and attributes in columns. The attributes are probe set identifiers, and the values of attributes are the picomolar concentrations of the spike-in material.

```
> dim(pData(spikein133))
[1] 6 42

> pData(spikein133)[1:3, c(1, 5, 10)]
           203508_at   204959_at   207777_s_at
Expt6_R1           2           4            16
Expt6_R2           2           4            16
Expt6_R3           2           4            16
```

The preprocessed expression intensities can be accessed using the pm() function. This submatrix is organized with *probes* in rows, and samples in columns:

```
> pm(spikein133)[c(1, 5, 10), 1:3]
       Expt6_R1   Expt6_R2   Expt6_R3
[1,]      245.0        238      238.0
[2,]     2325.0       2238     2591.0
[3,]      541.8        445      564.8
```

For cDNA platforms, containers named marrayRaw or RGList are used frequently. See the documentation of *marray* and *limma* packages for details.

*Processed Microarray Data.* Container design for corrected and normalized expression data emphasizes tight binding of experimental data and sample-level metadata, including probe identifiers and rich sample-level data. These containers have been implemented through preservation of manipulation idioms that are familiar through the use of simple data objects in pure R.

While it is possible to work with pure R matrices to represent gene expression experiments, Bioconductor enriches the data structure considerably. In the context of microarray data, let G denote the number of genes measured in a microarray experiment, let N denote the number of samples on which measurements were made, and let p denote a number of variables, such as treatment type, sample identifier, and sample characteristics, that identify important aspects of the experiment that should be known in any downstream analysis. Bioconductor defines a class of objects called exprSets that can represent all the relevant experimental data and metadata in a unified way. Specifically, if E is an instance of the exprSet class, then exprs (E) returns the GxN matrix of expression measures. pData(E) returns the

Nxp table of sample-level attributes, and description(E) returns a list of MIAME-defined experiment metadata attributes.

To illustrate this container concept, we interact with the `golubMerge` exprSet, which is supplied in the *golubEsets* package of Bioconductor. First we attach the data package and then we mention the 'golubMerge' exprSet, which combines the training and test data used in Golub *et al*. (1999).

```
> library(golubEsets)
> golubMerge
Expression Set (exprSet) with
  7129 genes
  72 samples
    phenoData object with 11 variables and 72 cases
  varLabels
    Samples: Sample index
    ALL.AML: Factor, indicating ALL or AML
    BM.PB: Factor, sample from marrow or peripheral
  blood
    T.B.cell: Factor, T cell or B cell leuk.
    FAB: Factor, FAB classification
    Date: Date sample obtained
    Gender: Factor, gender of patient
    pctBlasts: pct of cells that are blasts
    Treatment: response to treatment
    PS: Prediction strength
    Source: Source of sample
```

Note that this report about golubMerge data tells the number of genes and samples and provides details on the sample-level variables that are available. The exprSet can be treated as a ''two-dimensional'' object:

```
> sm <- golubMerge[1:3, 1:2]
```

This computes a new exprSet with three genes and two samples. All the appropriate sample level data are carried along.

Of particular interest are the numerical values of gene expression. This is obtained using the exprs() accessor function:

```
> exprs(sm)
                    [,1]    [,2]
AFFX-BioB-5_at      -342     -87
AFFX-BioB-M_at      -200    -248
AFFX-BioB-3_at        41     262
```

Sample-level data can be accessed very conveniently using a list accessor idiom:

```
> table(pData(golubMerge)$ALL.AML

ALL  AML
47   25
```

This gives the clinical leukemia classifications of the 72 patients.

*Metadata about Microarray Experiments.* The MIAME data model (Brazma *et al.*, 2001) provides an informal protocol for documenting microarray data sets in a uniform manner. The "MIAME" class has slots corresponding to the MIAME fields:

```
> getClass(''MIAME'')
Slots:

Name:    name            lab             contact     title
Class:   character       character       character   character

Name:    abstract        url             samples     hybridizations
Class:   character       character       list        list

Name:    norm-Controls   preprocessing   other
Class:   list            list            list

Extends: ''characterORMIAME''
```

A graphical user interface (GUI) for eliciting MIAME metadata can be run from R.

The phenoData class is used to manage sample-level metadata.

```
> getClass(''phenoData'')
Slots:

Name:          pData    varLabels    varMetadata
Class:    data.frame         list     data.frame
```

*General Biological Metadata.* The Bioconductor approach to biological annotation is somewhat complex, reflecting a variety of objectives that are difficult to harmonize simply. Some of the most prominent aims are as follow.

- To support substantive filtering of high-throughput data structures, allowing, for example, restriction of differential expression analysis to those probes that have been associated with specific molecular functions.
- To meld statistical analysis workflows with biological interpretation so that, for example, estimated contrast coefficients can be labeled with genome or pathway annotations as desired.
- To support visualization of assay data in meaningful genomic contexts, such as chromosomal location or pathway topology.

- To support stability of underlying annotation resources for statistical analyses that may take months to complete.

The data infrastructure meeting these objectives consists of collections of R environments. An example is the *GO* (Gene Ontology) package. Upon loading this package, executing the `GO()` function produces a listing of environments and information on their contents.

```
> GO()

Quality control information for GO
Date built: Created: Tue May 17 10:04:27 2005

Mappings found for non-probe based rda files:
  GOALLLOCUSID found 9287
  GOBPANCESTOR found 9529
  GOBPCHILDREN found 4765
  GOBPOFFSPRING found 4765
  GOBPPARENTS found 9529
  GOCCANCESTOR found 1536
  GOCCCHILDREN found 561
  GOCCOFFSPRING found 561
  GOCCPARENTS found 1536
  GOLOCUSID2GO found 62424
  GOLOCUSID found 7770
  GOMFANCESTOR found 7220
  GOMFCHILDREN found 1366
  GOMFOFFSPRING found 1366
  GOMFPARENTS found 7220
  GOOBSOLETE found 1020
  GOTERM found 18285
```

Data package environments are all named according to a convention. The environment name begins with the data package name and has a suffix indicating the specific contents. For example, GOBPANCESTOR is the environment that maps from GO identifiers to ancestors (generalizations) of the associated term in the Biological Process subontology.

In conjunction with the *annotate* package, high-level reports on environment contents can be extracted. The `lookUp` function takes an identifier token, the name of the data package of interest, and the suffix of the name of the environment to be searched.

```
> lookUp(''GO:0000001'', ''GO'', ''TERM'')

GOID = GO:0000001
Term = mitochondrion inheritance
Definition = The distribution of mitochondria, in-
  cluding the mitochondrial genome, into daughter
  cells after mitosis or meiosis, mediated by interac-
  tions between mitochondria and the cytoskeleton.
Ontology = BP
> lookUp(''GO:0000001'', ''GO'', ''BPPARENTS'')
              isa                      isa
''GO:0048308''          ''GO:0048311''
```

Accessing a description of the path(s) to the root of GO employs the environment GOBPANCESTOR.

The *GO* metadata package is a very general metadata resource, with information only about the gene ontology structure and content and, on some mappings, between gene catalogs and GO categories. Other metadata packages include

- *KEGG*—a series of environments providing information on the KEGG (Kyoto Encylopedia of Genes and Genomes) pathway catalog
- *cMAP*—environments that address the NCI Cancer Molecular Analysis Project unification of KEGG and BioCarta pathway and molecule catalogs
- *humanLLMappings*—environments that encode the mapping between Entrez Gene identifiers of human genes and other systems, such as UniGene clusters and GO categories
- *YEAST*—a collection of environments that map ORF identifiers to alias gene names, enzyme codes, PubMed entries, GO, and KEGG pathway catalog entries

*Workflows*

A hallmark of Bioconductor's approach to software design and dissemination is the support of user-constructed custom workflows. Because the software is provided in a loosely coupled system of packages, analysts can select and sequence tasks with great freedom.

Some developers have taken advantage of the component-based design to build unified graphical user interfaces. Prominent examples are *limmaGUI* and *affylmGUI*, which allow users to step from raw scanner outputs, through quality control and gene filtering, through linear modeling

for differential expression, to the development of hyperlinked lists of genes and associated annotations. Active discussion of further development occurs on the Bioconductor mailing list (https://stat.ethz.ch/mailman/listinfo/bioconductor).

### *Documentation Strategies*

The Bioconductor project recognized early on that broad utility would require a commitment to outstanding documentation resources. All Bioconductor software must be linked to manual pages that include executable examples. The project also developed a concept of ''vignette,'' which is a document that combines code, narrative, and graphics to illustrate an analysis process that may involve multiple packages. Vignettes can be processed by various software components to (1) export all code illustrated in the vignette computations, (2) transform code into an interactive graphical user interface so that the user can step through sequences of computations by pushing buttons and can evaluate effects of code execution in the current R environment, and (3) transform narrative, code, and graphics into PDF format documents. Bioconductor packages *annotate*, *DynDoc*, and *tkWidgets* coordinate the implementation of these functionalities.

## Array Preprocessing

### *Spotted Array Quality Control and Preprocessing*

Bioconductor includes a number of packages designed primarily for spotted array data. These include *marray*, *limma*, *vsn*, *arrayMagic*, which builds on *vsn*, and *arrayQuality*, which builds on *marray*. The primary raw data structures are the marrayRaw class of *marray* and the *RGList* class of *limma*; the *convert* package allows users to convert between them. Several functions read in raw data from each of several types of image quantitation programs. For example, for GenePix files, `read.marrayInfo` reads in target information; `read.Galfile` reads in the GAL files, and `read.GenePix` reads in the .GPR files.

Quality control (QC) is a first step. Microarray specialists can now recognize several kinds of common defects affecting individual spots, and also problems affecting whole regions of microarrays (Minor, 2006). Spot defects are often caused by printing problems; regional variations in ratios often reflect nonuniform hybridization or washing. We think the ability to examine microarray data using statistical QC measures provides an important safety net to uncover biases or artifacts in these data. The R statistical environment provides a number of general-purpose graphical tools for statistical QC, such as box plots for visualizing distributions at various time

points or for different batches of chips. The Bioconductor packages offer several special purpose QC tools adapted to dense microarray data. The *marrayRaw* class and the function `image( )` conveniently allow the user to display the spatial distribution of signals on spotted microarrays. This is illustrated by the following command, which produces an image of the green spot intensities for the third array of the data stored in the *marrayRaw* instance `raw.data`:

```
> image(raw.data[, 3], xvar = ''maGf'', bar = TRUE)
```

The *arrayMagic* package can display many different types of quality control plots. The package *arrayQuality* displays a number of common QC graphics in one web page. One of the pages produced by the command `maQualityPlots(raw.data)` is shown in Fig. 1.

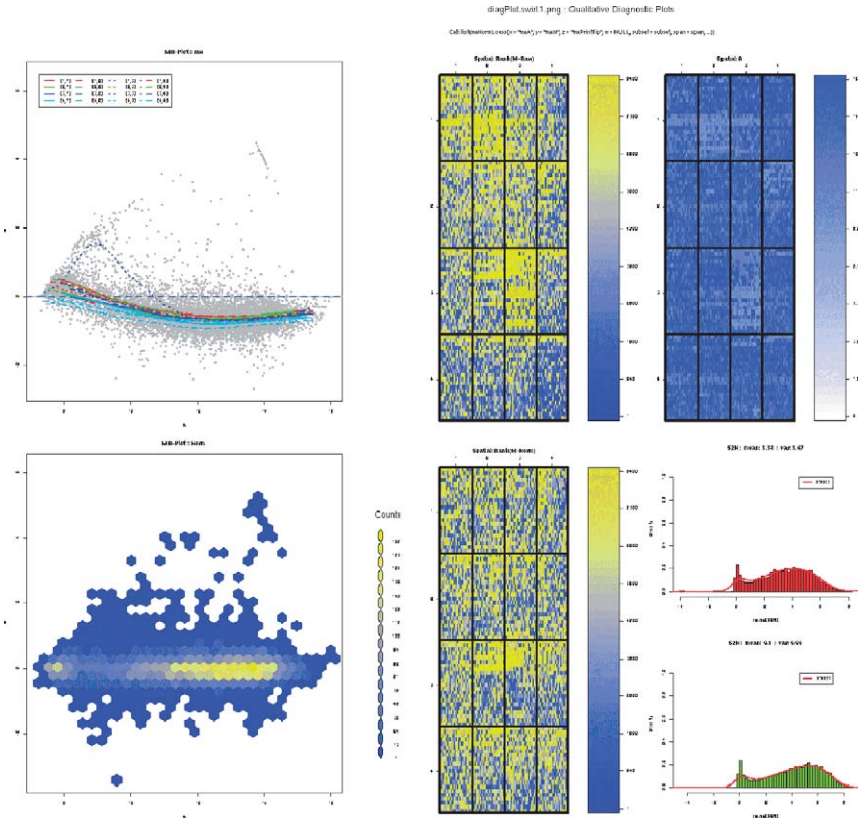


FIG. 1. Quality control plots from *arrayQuality*.

The plot shown in Fig. 1 shows the ratio-intensity (M-A) plot in the upper left, with separately colored loess traces for each print-tip group. The center shows two images of the ratios across the chip: before and after normalization.

The next step in analyzing spotted arrays is normalization. The marray classes offer options for both two-channel (with each array separately), and single-channel (between array) normalization. The command

```
> norm.data <- maNorm(raw.data, norm = ''l'')
```

produces the now widely used loess normalization of red-green ratios by compensating for an estimated intensity-dependent bias. Other options include location-scale normalizations, print-tip loess, and two-dimensional loess smoothing of ratios. A further step that adjusts ratios between arrays may improve replicability between arrays.

Further analysis of spotted microarray data can be done with functions in the *limma* package (and many of the preprocessing steps can be done entirely within *limma* or using the convert utilities to transfer data between formats). The *limma* package has especially good facilities for specifying design matrices for spotted array experiments (i.e., which samples are in which dye on which slides). The function `lmFit` uses design matrix information together with the ratios on all slides, to give the best consensus estimate of relative mRNA abundance in each sample. The function `eBayes` gives estimates of probability of differential expression for each gene.

*Preprocessing of Affymetrix Data*

The Affymetrix GeneChip poses inviting challenges for the biostatistician, and Biconductor incorporates a wealth of statistical thinking about Affymetrix outputs. The basis for all analysis is the *affy* package, particularly the methods for reading in and organizing information from Affymetrix raw data (CEL) files.

The Affymetrix GeneChip provides a number of probes for each target mRNA; these probes are distributed over the chip surface (see Dalma-Weiszhausz *et al.*, 2006). Affymetrix provides indexing information for probes on the array via the chip definition file (CDF) for each chip type. This information is made available to the affy package by R environments, which are a way of storing key value pairs.

To read in a set of CEL files in the current directory the user types

```
> cel.data <- ReadAffy()
```

Then to obtain the default expression measures (the RMA estimates), the user types

```
> expr.set <- rma(cel.data)
```

As for spotted arrays, quality control is vital for Affymetrix chips. Thus function is served by the nuse function in the *affyPLM* package and by new packages *harshlight* and *bias.display*. These packages produce false-color images displaying the discrepancies between individual probes on a chip, and the expected values of those probes, based on averaging across chips.

The affy package divides up the process of obtaining expression estimates into four steps: background correction, normalization, adjustment for nonspecific binding, and combining ratios from different probes. The basic affy package provides a number of options for each of these steps via the `expresso` function, and users may mix and match their favorite methods. Other more specialized packages offer some different variants on the multichip method.

The `expresso` function offers two methods for estimating and compensating background. The user specifies `bg.correct.method=''mas''` to compute a regional estimate of the lowest 2% of the probe intensities and to subtract them from the original values; this procedure follows the practice of Affymetrix' own MAS5.0 software. The `''rma''` method estimates empirically the density of nonspecific hybridization over the whole chip. Then it computes a Bayesian estimate of the specific hybridization for any individual probe by averaging over the distribution of possible nonspecific signals.

The issue of normalization is contentious. The affy package offers several options: setting `normalize.method = ''constant''` invokes a scaling transformation to bring the mean of all chips into agreement, as is done by Affymetrix' MAS5.0. The `''quantiles''` method computes an estimate of the distribution of all background-corrected signals and then shoehorns all individual distributions into that shape. There is active informal discussion of this normalization; some researchers feel that it is too strong, and they note that replicates for the most abundant genes are less consistent than by a simpler normalization. However, replicates for the majority of genes are more concordant, especially for the least abundant genes.

Much probe signal comes from nonspecific hybridization. The intent of Affymetrix was that the ''mismatch'' probes would provide a specific estimate of hybridization of similar but not identical cRNAs to the corresponding ''perfect match'' probes. It has been the experience of many statisticians that computations based on PM only give more consistent results (T. Speed, personal communication). However, the user may specify either option. The current thinking is that the relevant background signal for probe intensities is the nonspecific signal, and so steps 1 and 3 should be combined. This more sophisticated approach is implemented by the separate package *gcRMA*, which estimates nonspecific signals using the model developed in Zhang *et al.* (2004).

The final step in the `expresso` paradigm is synthesis of a single gene abundance estimate from the evidence of multiple probe signals. MAS5.0 constructs a measure by computing an average of corrected PM signals independently for each chip. The user may construct the MAS5.0 measure by specifying `summary.method = ''mas5''` in `expresso`. However, there is much information to be found by comparing across chips. A simple linear model for how the probe signal depends on gene abundance is that $S=af+e$, where $s$ represents signal, $a$ is gene abundance, and $f$ represents the affinity of a specific probe for its target gene; $e$ is noise. Raw data contain many outliers, and hence a robust fit is necessary. An adaptation of Tukey's median polish procedure provides a robust fit specified via the `summary.method = ''medianpolish''` option. A more sophisticated (and time-consuming) fit may be obtained by the function `fitPLM` in the `affyPLM` package.

### Addressing Multiple Comparisons

When searching among thousands of genes for evidence of differential expression, there are bound to be many genes that exceed even fairly rigorous $p$ value thresholds. Statisticians have developed several approaches to estimating and limiting the number of false positives. Two approaches that are implemented in Bioconductor are developing (1) more powerful genome-wide test statistics (*limma* and *siggenes*) and (2) methods for estimating the number of false positives (*multtest*) in a genome-wide test.

The basic idea behind moderated $t$ test statistics is that the $t$ statistic depends on an estimate of within-group variability. For small sample sizes (the usual case with microarray data), this variability estimate is itself highly variable; mistaken underestimates of within-group variability give rise to many false positives. However, by adjusting individual estimates of variation closer to a common value (such as their common mean), one can improve the majority of single estimates of variability, at the cost of introducing errors for a small number of genes.

The *siggenes* package implements ideas similar to those of the Statistics Applied to Microarrays program, whose approach was first described in Tusher and colleagues (2001).

The *limma* package implements an ''empirical Bayes'' method to estimating both variability and the probabilities that specific genes are expressed differentially. The user may choose a prior expectation of the number of changed genes (the conservative default is 1% of all genes on the array). Then the program returns a set of probabilities of differential expression for all genes.

The *multtest* package allows the user to compute several different estimates of the probability (or fraction) of false positives selected by a

statistical procedure. The most useful estimates are the family-wide error rate, the false discovery rate (FDR), and the tail probability of proportion of false positives (TPPFP). The family-wide error rate is the probability of any false positive being selected by the test procedure. Researchers commonly care about the typical confidence in a large list of genes. The FDR is an estimate of the expected proportion of false positives in a list (Benjamini and Hochberg, 1995). The TPPFP provides a probability bound for the fraction of false positives.

## Conclusions: Data Analysis for High-Throughput Biology and Bioconductor

In the coming decade, statistical methods will play a central role in dealing with the volume of data coming from high-throughput measurements such as microarrays. Through a collaborative process that is primarily informal, the Bioconductor project has engendered widely used software tools that address data translation and quality control, gene filtering, inference on differential expression, and phenotype prediction. The project has led to innovations in the production and dissemination of process-level documentation (vignettes and related dynamic documents) and in the important activity of reporting on statistical findings in biologically interpretable fashion, by allowing convenient binding of genomic, pathway, or functional ontology annotation to lists of features found to be statistically interesting, and by supporting straightforward creation and export of hypertext documents encoding these associations. The project has also spearheaded the use of a new object-oriented programming paradigm in R, the S4 system detailed in Chambers (1998).

Many open source software development and distribution projects have emerged in response to the challenges of the human genome project and to the excitement of postgenomic research agendas. These include projects focused on laboratory information management (BASE, Troein *et al.* (2006), `base.thep.lu.se`), scripting and programming language application (bioperl, biopython, biojava, bioruby, coordinated at open-bio.org), and data model and ontology development (open biological ontologies at obo.sourceforge.net, biopax at www.biopax.org).

The fundamental objectives of the Bioconductor project have been (a) promotion of advanced statistical technique in high-throughput biology and (b) reduction of barriers to interdisciplinary research. Through Bioconductor, statisticians have been given ready access to data examples and analysis practices in high-throughput biology. Biologists have been given access to the classic statistical analysis workflow components latent in R and also to emerging analysis strategies targeted directly at high-throughput

biology. This objective has been pursued under appropriate constraints of transparency (all software and data provided by the project are ''open source''). An extensive community of developers and users has emerged, united by an active mailing list and the software/documentation portal at www.bioconductor.org. We believe that the current situation of Bioconductor represents a partial achievement of the fundamental objectives. More work needs to be done to reduce the complexity of workflows, to help users match scientific needs to software capabilities, and to help developers integrate new techniques with existing structures. We are grateful to the Bioconductor Developers Core and to the many users and contributors who made this project possible.

## Acknowledgment

V. Carey's contributions are supported in part by NIH Grant 1R33 HG002708, a statistical computing framework for genomic data.

## References

Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J. R. Stat. Soc. Ser. B* **57,** 289–300.

Brazma, A., Hingamp, P., Quackenbush, J., Quackenbush, J., Sherlock, G., Spellman, P., Stoeckert, C., Aach, J., Ansorge, W., Ball, C. A., Causton, H. C., Gaasterland, T., Glenisson, P., Holstege, F. C., Kim, I. F., Markowitz, V., Matese, J. C., Parkinson, H., Robinson, A., Sarkans, U., Schulze-Kremer, S., Stewart, J., Taylor, R., Vilo, J., and Vingron, M. (2001). Minimum information about a microarray experiment (MIAME): Toward standards for microarray data. *Nature Genet.* **29,** 365–371.

Brazma, A., Kapushesky, M., Parkinson, H., Sarkins, U., and Shojatalab, M. (2006). Data storage and analysis in ArrayExpress. *Methods Enzymol.* **411,** 370–386.

Chambers, J. M. (1998). ''Programming with Data: A Guide to the S Language.'' Springer-Verlag, New York.

Dalma-Weiszhausz, D. D., Warrington, J., Tanimoto, E. Y., and Miyada, C. G. (2006). The Affymetrix GeneChip platform: An overview. *Methods Enzymol.* **410,** 3–28.

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A. J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J. Y., and Zhang, J. (2004). Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biol.* **5,** R80URL:http://genomebiology.com/2004/5/10/R80.

Golub, T. R., Slonim, D. K., Tamayo, P., Slonim, D., Golub, T. R., and Kohane, I. S. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science* **286,** 531–537.

Ihaka, R., and Gentleman, R. (1996). A language for data analysis and graphics. *J. Comput. Graph. Statist.* **5,** 299–314.

Minor, J. M. (2006). Microarray quality control. *Methods Enzymol.* **411,** 233–255.

Troein, C., Vallon-Christersson, J., and Saal, L. H. (2006). An introduction to BioArray Software Environment. *Methods Enzymol.* **411,** 99–119.

Tusher, V. G., Tibshirani, R., and Chu, G. (2001). Significance analysis of microarrays applied to the ionizing radiation response. *Proc. Natl. Acad. Sci. USA* **98,** 5116–5121.

Zhang, L., Miles, M. F., and Aldape, K. D. (2004). A model of molecular interactions on short oligonucleotide microarrays: Implications for probe design and data analysis. *Nature Biotechnol.* **21**(7), 818–821.

# [9] TM4 Microarray Software Suite

*By* ALEXANDER I. SAEED, NIRMAL K. BHAGABATI, JOHN C. BRAISTED, WEI LIANG, VASILY SHAROV, ELEANOR A. HOWE, JIANWEI LI, MATHANGI THIAGARAJAN, JOSEPH A. WHITE, and JOHN QUACKENBUSH

## Abstract

Powerful specialized software is essential for managing, quantifying, and ultimately deriving scientific insight from results of a microarray experiment. We have developed a suite of software applications, known as TM4, to support such gene expression studies. The suite consists of open-source tools for data management and reporting, image analysis, normalization and pipeline control, and data mining and visualization. An integrated MIAME-compliant MySQL database is included. This chapter describes each component of the suite and includes a sample analysis walk-through.

## Introduction

The Human Genome Project was envisioned as a grand endeavor that would change biology by providing a catalog of genes in humans and other model organisms. Although a large number of genome sequencing projects, including that of the human genome, have been declared finished, the collection of the sequence itself has not fundamentally altered our approach to understanding biological systems. Rather, it has been the development of techniques and technologies that allow us to analyze patterns of expression for sets of genes, proteins, or metabolites approaching the total number that are active in an organism at any given point in time.

Since their introduction in 1995 (Lipshutz, 1995; Schena, 1995), DNA microarrays have matured significantly to become the most widely used technique for the analysis of global patterns of expression and represent a technology that is now used routinely as a means of generating testable hypotheses prior to other studies. DNA microarrays consist of an arrayed

METHODS IN ENZYMOLOGY, VOL. 411 0076-6879/06 $35.00

DOI: 10.1016/S0076-6879(06)11009-5