2

Visualization and Annotation of Genomic Experiments

Robert Gentleman
Vincent Carey

Abstract

We provide a framework for reducing and interpreting results of multiple microarray experiments. The basic tools are a flexible gene-filtering procedure, a dynamic and extensible annotation system, and methods for visualization. The gene-filtering procedure efficiently evaluates families of deterministic or statistical predicates on collections of expression measurements. The expression-filtering predicates may involve reference to arbitrarily complex predicates on phenotype or genotype data. The annotation system collects mappings between manufacturer-specified probe set identifiers and public use nomenclature, ontology, and bibliographic systems. Visualization tools allow the exploration of the experimental data with respect to genomic quantities such as chromosomal location or functional groupings.

2.1 Introduction

The Bioconductor project (`www.bioconductor.org`) was initiated at Dana Farber Cancer Institute to gather statisticians, software developers, and biologists interested in advancing computational biology through the design, deployment, and dissemination of open-source software. The R statistical computing environment (`www.r-project.org`) is the central computing and interaction platform for tools created in Bioconductor. Bioconductor's software offerings consist of R packages and data images to support many aspects of computing and inference in bioinformatics. Key packages that will be reviewed in this chapter include:

- `Biobase`: the basic data structures and algorithms required for storage and exploration of genomic, phenotypic, and gene expression data obtained in microarray experiments;

- `genefilter`: routines for efficient identification of genes satisfying arbitrarily complex, statistically defined conditions;
- `edd` (expression density diagnostics): algorithms for evaluating the diversity of gene expression distributions in phenotypically defined cohorts;
- `annotate`: programs and data structures for connecting genomic and phenotypic data to biological and clinical annotation and literature to facilitate interpretation and hypothesis-driven modeling;
- `geneplotter`: programs to assist in the visualization of experimental results.

Other Bioconductor/R packages are described in Dudoit and Yang (Chapter 3, this volume) and Irizarry et al. (Chapter 4, this volume).

After describing and formalizing the motivations and choices of data structures and algorithms for this project, we will illustrate the use of these software tools with two large gene expression databases. The first is the set of 47 U68 Affymetrix arrays discussed in Golub et al. (1999). The data are available in their original form at `http://www-genome.wi.mit.edu`. The samples were collected on individuals with acute myelogenous or acute lymphocytic leukemia. The second dataset is a collection of 89 U95 arrays from Zhang, Derdeyn, Gentleman, Leykin, Monti, Ramaswamy, Wong, Golub, Iglehart and Richardson (2002) with additional data on HER2 status provided by Dr. A. Richardson. These data are samples collected on metastatic and nonmetastatic breast cancer tumors. Both datasets and transcripts of data analysis sessions related to this chapter are available at `www.bioconductor.org/Docs/Papers/2002/Springer`.

2.2 Motivations for Component-Based Software

Microarray experiments are based on collections of tissue samples (usually fewer than 100) on which expression of messenger RNA (mRNA) has been measured.

For concreteness, we focus on the analysis of oligonucleotide arrays. Let $i = 1, \ldots, I$ index independent microarrays. The ith array supplies x_{ji}, $j = 1, \ldots, J$ expression measures. There is usually substantial processing of the raw experimental data needed to obtain these expression values. The steps involved in this processing are discussed in Irizarry et al. (Chapter 4, this volume) and Li and Wong (Chapter 5, this volume). We assume that the expression values are comparable across arrays although not necessarily across genes.

The large number of genes J (usually $J > 1000$ and often $J > 10,000$) places special requirements on computational and inferential tools for the analysis of microarray experiments. New methods of computation and inference are also required for the complex task of connecting numerically detected patterns of gene expression and resources, typically textual in na-

ture, that allow biologic interpretation of these patterns. We now discuss details of some of the basic motivations for establishing flexible component-based approaches to filtering, annotation, and visualization of gene expression data.

Diversity and dynamic nature of microarray formats and outputs. A microarray generally consists of a spot (or a set of spots) on which specific portions of mRNA should cohybridize. The number of spots, the length of the sequence against which cohybridization is performed, and the nature of the sequence depend on the manufacturer of the chip. The two most popular techniques are cDNA arrays as described by Shalon et al. (1996) and Affymetrix short oligomer arrays. It is important to realize that there are limitations to the inferences that can be derived from these arrays. In both cases, we can measure the amount of mRNA that hybridizes to the spot(s) on the array. Biologic or clinical inference relies on the imperfect and evolving mapping from the EST to the gene. This mapping should be carried out on data that are as recent as possible, so software implementations must avoid "hard-coding" any but the most permanent features of this mapping. Ideally, the software will make real-time use of Web-based repositories that satisfy given requirements for currency and accountability.

Fallibility of EST construction. Most arrays use probes that are based on expressed sequence tags (ESTs). We will refer to the probes and their targets incorrectly but interchangeably as ESTs and genes. An EST is a short (typically 100–300 bp) partial cDNA sequence. cDNA is DNA synthesized by the enzyme reverse transcriptase using mRNA as a template. In practice, cellular mRNA is used together with reverse transcriptase to build cDNA. The resulting cDNA sequences are then used to build ESTs. This process is not infallible, and some proportion of the ESTs will be incorrect (or incorrectly labeled).

Need to distinguish transcript abundance from gene activity and protein abundance. The expression of the genetic information contained in DNA occurs in two stages. The first is transcription, where DNA is transcribed into RNA. The second stage is translation, where the RNA is translated into a protein. The central dogma of molecular biology is that DNA makes RNA makes protein. Microarray technologies measure levels of mRNA in different samples. Thus, they measure transcript abundance, which may or may not relate to the presence of a gene (since the gene need not be transcribed). Transcript abundance may or may not relate to the presence of a protein (since the mRNA need not be translated). Interpretation of microarray outputs may also be affected by translocations or increases in copy number that may be present in only a fraction of samples with a given phenotype. Either increased copy number with the usual expression control or increased/enhanced expression with the copy number left the same can result in increased mRNA expression.

Need to separate potentially functional ESTs from housekeeping sequences. For any given tissue in the human body, it is estimated that about 40% of the genome is expressed. Another portion of the genes measured will have a relatively constant transcript abundance across samples. These genes may be performing *housekeeping activities* or other activities that are unrelated to the processes being studied. Therefore, in most cases there are a relatively large number of ESTs that are inherently uninteresting, and some means of removing them from the remainder of the analysis will be helpful. Many of the tools that we employ are computationally expensive, and any reduction of the data will be rewarded by decreased analysis times. One approach is to filter out the uninteresting genes so that attention can be focused on those genes that have some potential to be interesting.

Necessity and limitations of gene-specific processing. With Affymetrix arrays, comparisons between expression levels estimated from different probe sets should be made with caution. For these arrays, the intensity may be affected by the probe sequences used, and two different probe sets may have greatly different estimated expression values when in fact the abundances of the mRNAs are quite similar. With cDNA arrays, provided all arrays have been hybridized with a common baseline, a comparison between genes is probably valid. Gene-at-a-time processing is a severely limited analytic framework and is incapable of great fidelity to the biology involved. The systems that we are studying are complex, and there are always interactions between different gene products. There are many systems that are more complex. For example, the ratio of BAX (BCL2-associated X protein) to BCL2 determines the cell's fate. If the level of BCL2 is larger than that of BAX, then apoptosis (programmed cell death) is suppressed. When the level of BAX is larger than that of BAD, apoptosis is promoted (Helmreich, 2001; p. 241). Hence, we are not interested in changes in one of these two genes but rather in changes in their ratio.

The considerations just enumerated have played a significant role in shaping our design of Bioconductor software components. We put a high premium on designs that allow adaptation to new approaches and resources—at both the biotechnologic and inferential levels—that help overcome the limitations and ambiguities inherent in the current state of the art of microarray experimentation and interpretation. Formalization of the key resources of our approach is provided in the next section.

2.3 Formalism

Recall that $i = 1, \ldots, I$ indexes the samples or chips, and $j = 1, \ldots, J$ indexes genes that are assumed to be common across all chips. For sample i, a q-vector y_i contains phenotypic and/or demographic data on the patients from which samples were derived, such as age, sex, disease status, duration of symptoms, and so on. These data are conceptually and often adminis-

tratively distinct from the expression data (which may be managed in a completely different database). Labels $s_i = s(y_i)$ are available to classify samples; for example, $s_i \in \{normal, diseased\}$. Note that we will commonly use the term *phenotypic data* very loosely to refer to any nongenomic data related to tissue samples or their donors.

An *experiment metadata structure* is a description of conditions and materials used in a biological experiment. Standard specifications of metadata structures have been proposed (e.g., MIAME `http://www.mged.org/Workgroups/MIAME/miame.html`; Brazma et al., 2001). Our framework is sufficiently broad to accommodate such annotation.

An *annotation structure* is a mapping between EST identifiers and standardized nomenclatures for associated genes. The mapping may be formalized as a set of ordered sequences with specified positions for conventional (e.g., manufacturer-defined) EST identifiers and associated standard nomenclature tokens for the genes related to that EST.

A *microarray experiment database* is the ordered quadruple $S = \langle X, Y, A, M \rangle$, where X is a $J \times N$ matrix with columns representing J-vectors of gene expression measurements on each of N tissue samples, Y is an $N \times p$ matrix with rows representing p-vectors of phenotypic information on the N tissue samples or their donors, A is an annotation structure, and M is an experiment metadata structure. The floating-point number x_{ji} is the (j, i) entry of matrix X, representing the measured expression level for gene j on tissue sample i. The datum y_{iq} is the value of phenotypic variable q on tissue sample i or its donor. x_j is the N-vector of expression values for gene j over all donors.

A *gene filter element* is a Boolean-valued function $f(x, Y)$ of an N-vector of expression values x and an $N \times p$ phenotype matrix Y. An example is a function that uses the phenotype data to divide x into two samples (e.g., diseased and nondiseased) and returns TRUE if and only if a two-sample test of location shift rejects the null hypothesis of a common location for the two samples.

A *gene filter* is a collection of gene filter elements that can be applied to a microarray experiment database. Let F denote such a collection. Then, the action of a gene filter on a microarray experiment database S is to define the index set $I_{S,F} = \{j = 1, \ldots, J | f(x_j, Y_S) = 1, \text{ all } f \in F, x_j \in X_S\}$, where X_S and Y_S are the expression and phenotype components of S.

2.4 Bioconductor Software for Filtering, Exploring, and Interpreting Microarray Experiments

2.4.1 Formal Data Structures and Methods for Multiple Microarrays

The choice and design of data structure to represent multiple microarray experiments is one of the most critical processes of software development

in this domain. Regimentation of the structure is important to establish trustworthiness of related computations and to facilitate reuse of software components that effectively navigate and process elements of the structure. Flexibility of the structure is also of great importance so that changes in biotechnology and research directions do not necessitate complete redesign. Our approach to data structure design for this problem preserves the conceptual independence of genomic and phenotypic information types and exploits the *formal methods and classes* in the R package `methods` which was developed by Chambers (1998).

The `methods` package provides a substantial basis for object-oriented programming in R. Object-oriented programming is a well-established approach to dealing with complex data structures and algorithms. When we can conceive of our data as an object with a well-defined set of properties and components of various types, then an object-oriented programming system may be used to represent our data as an instance of a class of similarly structured objects. Analysis algorithms defined in the object-oriented system are freed from responsibility for checking the structure and contents of the components of objects instantiating the class. Furthermore, when proper design and deployment discipline is maintained, it is often possible to extend the structure of an object without affecting the behavior of software developed for previous versions of the object. Thus, the object-oriented approach gives access to the regimentation and flexibility required for the creation and support of durable software for bioinformatic inference.

Classes `exprSet` and `phenoData`

To coordinate access in R to the genomic and phenotypic data generated in a typical microarray experiment, we defined two classes of objects. One is called the `exprSet` class and the other the `phenoData` class.

The `phenoData` class was designed to hold the phenotypic (or sample level) data. Objects of this class have the following properties or *slots*:

pData A `data.frame` with samples as the rows and the phenotypic variables as the columns.

varLabels A list with one element per phenotypic variable. Names of list elements are the variable names; list-element values are character strings with brief textual descriptions of the associated variables.

An instance of class `exprSet` has the following slots:

exprs An array that contains the estimated expression values, with columns representing samples and rows representing genes.

se.exprs An array of the same size as `exprs` containing estimated standard errors. This may be NULL.

phenoData An instance of the `phenoData class` that contains the sample level variables for this experiment. This class is described above.

description A character string describing the experiment. This will probably change to some form of documentation object when and if we adopt a more standard mechanism for documentation.

annotation The name of the annotation data that can be used for this `exprSet`.

notes A character string for notes regarding the analysis or for any other purpose.

Once a class has been defined, instances of it can be created. A particular dataset stored in this format would be called an instance of the class. Although we should technically always say something like *x is an instance of the* `exprSet` *class*, we will often simply say that *x is an* `exprSet`.

Formal Methods for `exprSet`s

A number of methods for the `exprSet` and `phenoData` classes have been implemented. The reader is directed to the documentation in our packages for definitive descriptions. Here, we illustrate the various methods with examples based on a celebrated collection of array experiments. The `golubEsets` package includes `exprSets` embodying the leukemia data of Golub et al. (1999). Upon attaching the `golubTrain` element of the package, we may invoke the following `exprSet` methods:

Show the data. The generic `show` function (invoked upon mention of an `exprSet` instance) gives a concise report.

```
> golubTrain
Expression Set (exprSet) with
  7129 genes
  38 samples
   phenoData object with 11 variables and 38 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
    Sex: Factor, sex of patient
    pctBlasts: pct of cells that are blasts
    Treatment: response to treatment
```

```
    PS: Prediction strength
    Source: Source of sample
```

Subset of genes. Subscripting on the first index returns a matrix of expression values of the corresponding genes. Notice that the row names are Affymetrix identifiers for the ESTs used. The `annotate` package will use these identifiers to map to different quantities such as the LocusLink identifiers or chromosomal location. Affymetrix uses the convention that identifiers that begin with `AFFX` are used for quality-control purposes and are generally uninteresting for any analysis. One may want to remove these before analyzing the data.

```
> golubTrain[1:4,]
                [,1] [,2] [,3] ...
AFFX-BioB-5_at  -214 -139  -76
AFFX-BioB-M_at  -153  -73  -49
AFFX-BioB-3_at   -58   -1 -307
AFFX-BioC-5_at    88  283  309
```

Extract phenotype vector. This method exploits a specialization of the `$` operator to deal with instances of the `exprSet` class.

```
> table(golubTrain$ALL.AML)

ALL AML
 27  11
```

Subset of patients. Subscripting on the second index returns a matrix of expression values of the corresponding tissue donors. In this example, we obtain the dimensions after restriction to patients whose `phenoData` indicates that they have acute lymphocytic leukemia.

```
> print(dim(golubTrain[
        ,golubTrain$ALL.AML=="ALL"]))

[1] 7129   27
```

Accessor functions. A number of functions are provided to allow access to more primitive representations of expression data. The `exprs` function applied to an `exprSet` returns the $J \times N$ matrix of expression results. The `pData` function returns the $N \times p$ `data.frame` of phenotypic data.

Before concluding this topic, we make some remarks on the problem of concisely specifying `exprSet` subsets. Familiar syntax has been established to allow reference to subsets of collections of arrays. A key obligation is

to keep the genomic and phenotypic data correctly aligned, so, if `eS` is an instance of an `exprSet`, we may ask for `eS[,1:5]`. The second index in this subscripting expression refers to tissue samples. The value of this expression is an `exprSet` whose contents are restricted to the first five tissue samples in `eS`. The new `exprSet`'s `exprs` element contains the first five columns of `eS`'s `expr` array. The new `exprSet`'s `se.exprs` array is similarly restricted. But the `phenoData` must also have a subset made and for it we want the first five *rows* of `eS`'s `phenoData` dataframe since it is in the more standard format where columns are variables and rows are cases.

Other tools for working with `exprSet`s are provided, including methods for iterating over genes with function application and sampling from patients in an `exprSet` (with or without replacement).

2.4.2 Tools for Filtering Gene Expression Data: The Closure Concept

We have seen that the `golubTrain` `exprSet` includes results for 7129 genes. The `genefilter` package of Bioconductor provides a collection of functions allowing computationally efficient reduction of the set of genes of interest. The user interface is the `genefilter` function, which accepts a $J \times N$ matrix of expression data and an instance of class `filterfun`. The `filterfun` class supports the combination of functions to define a sequence of filtering criteria. In the current formulation, the `genefilter` package supports *marginal* filtering: criteria are constructed and evaluated to retain or exclude a gene based solely on features of the distribution of expression values of that gene (in relation, of course, to the phenotypic features of the samples on which gene expression was measured). This is to be distinguished from *joint* filtering, in which the retention or exclusion of a gene may depend on distributions of other genes. An example of joint filtering is given in subsection 2.6.1.

We have designed the `genefilter` package to support very flexible specifications and combinations of filtering criteria. Suppose that we wish to restrict attention to genes in `golubTrain` that have expression values exceeding 100 for at least five donors and for which the coefficient of variation is at least 2. The following commands carry this out.

```
myAbsLB <- kOverA( k=5, A=100 ) # absolute lower bound
myCVspec <- cv( a=2 )          # lower bound on CV
myFilterSeq <- filterfun( myCVspec, myAbsLB )
myInds <- genefilter( exprs(golubTrain), myFilterSeq )
```

Now

```
Train2 <- golubTrain[myInds,]
```

has 1534 genes. To introduce more stringent filtration, arguments to the filter components can be altered and the process of building and applying

the filter collection `myFilterSeq` can be repeated, or, new filters could be constructed and applied to `exprs(Train2)`.

The command

```
myAbsLB <- kOverA( k=5, A=100 )
```

defines a *closure*. This is an R function accompanied by an environment defining local data. The limit, `A`, and number of exceedents required, k, are bound to 5 and 100 in `myAbsLB`, which is applied to all N-vectors of expression values for all genes in `exprs(golubTrain)`. The use of closures allows us to provide a simple but very flexible interface. Arbitrarily many filters can define a filtration. Each of the filters may make use of user-specified parameters or phenotypic data structures. For more background on closures in R, see Gentleman and Ihaka (2000).

This approach can be contrasted with one where long lists of optional parameters are supplied. Our experience suggests that the latter is error-prone. Separation of the task of identifying and creating specific filters from the task of applying those filters provides a substantial simplification.

2.4.3 Expression Density Diagnostics: High-Throughput Exploratory Data Analysis for Microarrays

A basic problem in gene discovery exercises is the formulation of test procedures that powerfully discriminate distinct patterns of gene expression in phenotypically distinct tissues. In standard statistical applications, exploratory data analyses using graphics and goodness-of-fit appraisals are used to match the test procedure to the characteristics of the data. For example, data from long-tailed distributions will often be log-transformed or scrutinized for outliers. The magnitude and relative complexity of gene expression array datasets has been a hindrance to exploratory analysis and investigation of test selection procedures based on characteristics of gene expression distributions across samples. If gene expression distributions are highly diverse in shape, discovery procedures will need to adapt to the shapes present in order to possess reasonable power.

As a first step toward more adaptive gene discovery methods, the `edd` package performs a species of high-throughput exploratory data analysis. To overcome the difficulty of evaluating thousands of histograms or density smooths to assess skewness, multimodality, outlier-proneness, or other departures from Gaussianity, we compare gene- and stratum-specific empirical distribution functions for expression data to a catalog of specified reference distributions, labeling each gene with the closest matching reference distribution or "doubt." This allows the grouping of genes into broad classes to facilitate more focused diagnostic analysis or discriminative testing. We now provide some formal definitions of components of the procedure. We suppose that the underlying distributions are of continuous type for all

genes. Expression distribution classification involves three steps.

Location and scale reduction. Define $m_F = F^{-1}(1/2)$ to be the median of the cumulative distribution function (cdf) F and $a_F = c \cdot \text{median}_{x \sim F}|x - m_F|$ to be the scaled median absolute deviation (MAD) of F. Scaling factor $c = 1.483$ is used to obtain consistency of MAD for σ at $N(\mu, \sigma^2)$. The centered and scaled cumulative distribution function (cdf) corresponding to an arbitrary cdf F is defined as

$$F^*(y) = \Pr(a_F^{-1}[Y - m_F] < y).$$

We will use the asterisk superscript to denote cdfs or deviates from distributions that have been subjected to this transformation. For example, $X^* \sim \text{Beta}^*(r,t)$ implies that $X^* = s^{-1}[X - m]$, where $X \sim \text{Beta}(r,t)$ and m(resp. s) are the median (resp. MAD) of the Beta distribution with parameters r and t. The function `centerScale` in `edd` can be applied to any real vector X to obtain X^*.

Formation of reference catalog. We develop the tools to simulate from and evaluate quantiles of a collection of location- and scale-reduced parametric distributions: Beta*(2,8), Beta*(8,2), χ_1^{2*}, LN*(0,1) (log normal with mean zero and standard deviation unity after log transformation), MIX$_p^*$, where MIX$_p$ denotes the two-component mixture $pN(0,1)+(1-p)N(4,1)$, t_3^*, and $U^*(0,1)$. Other catalog elements can be added as needed. These candidates were chosen to represent a diversity of shapes and to allow some assessment of the possibility of confusion (χ_1^{2*} and LN*(0,1) are very similar shapes). In general, there will be R location- and scale-reduced reference distributions in the catalog, which we will denote $\mathbb{H} = \{H_1, \ldots, H_R\}$. The functions `makeCandmat.raw` and `makeCandmat.theor` create reference catalogs based on simulation and quantile calculation after location and scale reduction, respectively.

Classification. The empirical distribution function (EDF) of location- and scale-reduced expression observations on each gene is treated as an I-dimensional multivariate datum. For gene g (in a stratum s), this EDF is denoted $\hat{F}_{sg}$; the stratum index will be suppressed unless required. Three approaches to classification of gene-specific EDFs are investigated:

- *Nearest-neighbor classification.* A prespecified number c of I-dimensional reference candidates are simulated from each element of $\mathbb{H}$. This leads to $c \cdot R$ I-vectors that serve as a training set, each with known shape. k-NN classification is conducted for each $\hat{F}_g$ with parameters k (number of neighbors to be polled) and l (minimum number of concordant votes of type s required to classify the candidate into category s). The expression distribution of gene j is declared to be of shape $r \in \{1, \ldots, R\}$ if l of the k closest catalog members are simulated from H_r and is declared to be of unknown shape otherwise.

- *Test-based classification.* For each j and each $s \in \{1, \ldots, R\}$, compute the p-value p_s of the Kolmogorov–Smirnov statistic testing the hypothesis that $x_j \sim H_s$. The expression distribution of gene j is declared to be of shape r if p_r exceeds a specified lower bound and is uniquely the largest of the p_s and is declared to be of unknown shape otherwise.
- *Model-based classification.* The $c \cdot R$ I-dimensional training set described under the nearest-neighbor approach is used to construct a neural network that predicts class membership or "doubt" for each $\hat{F}_g$.

The function `edd.unsupervised` includes options for the type of classification algorithm to be used and can be applied to `exprSet`s. It returns a set of J classification labels indicating for each gene in the `exprSet` the best-fitting element of the reference catalog.

2.4.4 Annotation

Relating the ESTs comprising a class of arrays to various biological data resources (e.g., genomic maps, protein function characterizations, general literature of clinical genetics) is an essential part of the analysis. If an analysis is primarily data-driven, one needs to relate the numerical patterns discovered among ESTs and phenotypic conditions to information about gene structure and function. If an analysis is hypothesis-driven, one needs to use information on gene structure and function to restrict or otherwise structure the sets of ESTs and phenotypes considered in the analysis to conform to the hypotheses of interest. These activities are problematic because biological and clinical data resources are continually changing, as is our understanding of their contents and how they are to be harvested and used. Based on these observations, we have chosen a mechanism for capturing and supplying annotation data that is flexible and easily updated.

We have divided the process into two components. One component consists of building and collating annotation data from public databases. The second component is the extraction and formatting of the collated data into a format suitable for end users. Data analysts can simply obtain a set of annotation tables for the set of ESTs that they are using. We will consider only this use of the data in this discussion. This is a rapidly changing field, so readers should consult the Bioconductor Web site for current details on the system's resources and methods.

Our aim is to provide data structures facilitating selection of genes according to certain features or a priori conventional classifications such as membership in functional groups, involvement in biological processes, or chromosomal location. Currently, annotation structures are available from the Bioconductor project in a variety of formats. Datasets have been marked up as XML files with explicit DTDs and have also been formatted as R objects suitable for loading into R using the `load` function. Internally (to R), they are managed using hash tables, and hence we require

a unique key for each entry. Mapping from the EST (or manufacturer's identification) is then fairly straightforward.

Affymetrix Inc. produces several chips for the human genome. The most popular in current use are the U95v2 chips (with version A being used most often). The probes that are arrayed here have Affymetrix identifiers. We provide functions that map the Affymetrix identifiers to a number of other identifiers such as those provided by LocusLink and GenBank.

In addition, we have assembled relations between the different identifiers and their Gene Ontology (GO) values. See `http://www.geneontology.org`. GO is an attempt to provide standardized descriptions of the biological relevance of genes into the three categories *biological process*, *cellular component*, and *molecular function*. Within a category, the set of terms forms a directed acyclic graph. Genes typically get a specific value for each of these three categories. We trace each gene to the top (root node) and report the last three nodes in its path to the root of the tree. These top three nodes provide general groupings that may be used to examine the data for related patterns of expression. Using the GO annotation allows users to consider subsets of genes for analysis using groupings under any of these headings. syntenic region, and sets of orthologs.

Often, researchers are interested in finding out more about their genes. For example, they would like to look at the different resources at NCBI. HTML pages with active links to the different online resources can easily be produced. The function `ll.htmlpage` is one example of a function designed to provide links to the LocusLink Web page for a list of genes.

It is also possible using connections and the XML package to open http connections and read the data from the Web sites directly. The resulting text could then be processed using other R tools. For example, if abstracts or full-text searchable articles were available, these could be downloaded and searched for relevant terms. Examples will be provided in Section 2.6 below.

2.5 Visualization

Visualization of data is an important and potentially very powerful method for exploring and understanding data. Like all other techniques for the analysis of genomic data, visualization is in its infancy. We consider just a few plots that might be useful. We demonstrate how these plots help data analysts understand their data.

Perhaps the most used visualization aid for microarray data are the *heat maps*, or displays of genome-wide expression patterns, proposed by Eisen et al. (1998). These plots generally cluster together both genes and samples that have similar expression levels. The argument for doing this is that genes with similar levels of expression often have a similar function. Finding genes that have a similar function is often one of the things in which we are interested.

A typical rendering is as a rectangular region with rows given by ESTs and columns by samples. Each small rectangle is colored to indicate the expression level of the specific EST for the specific sample. Expression level is usually indicated by both intensity and color (often red for high and green for low, which is rather unfortunate for the color-blind). Usually, some form of hierarchical clustering is performed to arrange the rows and the columns so that there are relatively large homogeneous regions of red and green. Rather than examine these rather well-known plots—easily constructed in R using the `image` function and a suitable set of colors—we will provide a few suggestions for other types of graphics that might be interesting.

2.5.1 Chromosomes

One might be interested in locating where a gene or set of genes is located in the genome. The chromosomal locations can be obtained from the `annotate` package and can then be used to construct visual tools. In this section, we demonstrate some of the tools available in the `geneplotter` package.

Different genomes have different numbers of chromosomes, each of different length. Additionally, chromosomes are generally double stranded, and a gene can be encoded on either strand (one strand is called the plus strand and the other the minus strand). We generally give an indication of which strand a gene is on in all visualizations.

Plots of expression and its association with chromosomal location might be helpful. The function `alongChrom` in the `geneplotter` package provides plots of expression level along the chromosomes. There are several options available. One that seems initially promising is to plot cumulative expression along a chromosome or region of a chromosome. Large jumps in this plot indicate high levels of expression, whereas flat spots indicate a lack of expression.

An alternative to plotting the cumulative expression level along a chromosome is to plot individual levels at the appropriate position. There are a variety of transformations that might be useful, such as the z-transformation $(x - \bar{x})/\mathrm{sd}(x)$ or transforming to ranks.

In breast cancer, one of the important prognostic factors is the presence of the ERBB2 gene. This particular genomic region is subject to amplification through duplication. Increased expression of ERBB2 is often associated with an increased copy number, that is, a region of the chromosome containing ERBB2 is replicated. There can be many replicates of the region contained in the genome, and there can be many genes on each replicate. A region of this type is often referred to as an amplicon. The amplicon can be of a different size in different patients.

Using `annotate` the probes for ERBB2 on the HgU95A chip were identified as `1802_s_at`, `1901_s_at`, and `33218_at`. A `pairs` plot for the log of the expression data for these three probes is given in Figure 2.1.

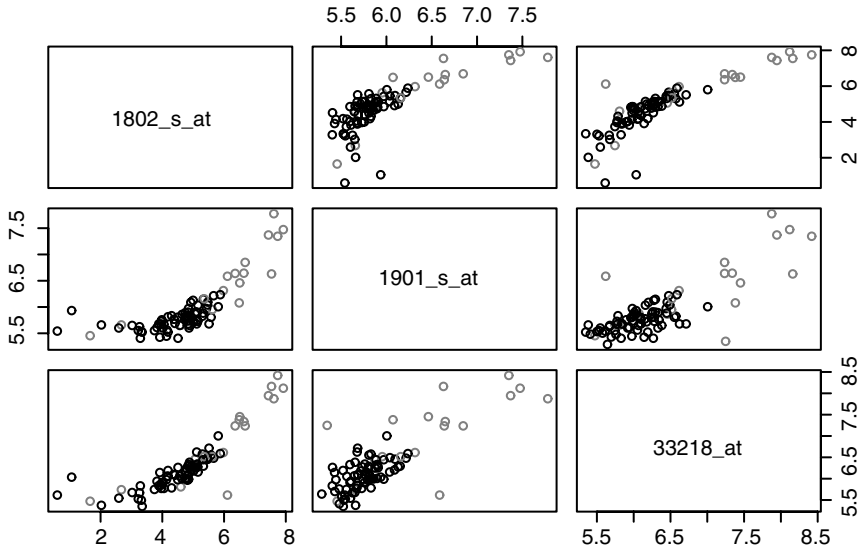

FIGURE 2.1. A pairs plot of the expression data for probes for the gene ERBB2.

We next locate ERBB2 using the data available in the `annotate` package. It is located (approximately) at position 41940228 on the plus strand of Chromosome 17. Thus, we next examine Chromosome 17 more closely. The first plot, Figure 2.2, is simply the cumulative expression of genes on Chromosome 17 by strand. In this case, the plot was not particularly informative, and a more detailed look at the region of the amplicon seems justified.

An examination of the relevant literature by Dr. A. Richardson implicated `32064_at`, PPARBP; `37355_at`, MLN64; and `1680_at`, GRB7 as genes that might also be included in the amplified region, or *amplicon*. To better see their effects, we now only plot expression levels for that region of Chromosome 17. Once again, we use the `alongChrom` function. The plot region is restricted to the portion of Chromosome 17 between 41800000 and 42000000 bases. The cumulative expression (ERBB2-positive patients are colored red) is provided in Figure 2.3. Now we can easily see the high levels of expression of the genes (for ERBB2-positive patients) on both strands of the chromosome.

In these plots, we have chosen to plot the genes equally spaced. There are a great number of options available for `alongChrom`, and the reader is encouraged to explore them.

Another question that one might like to ask about these genes is whether the expression levels of different genes are correlated (across subjects). To explore this, one could look at `pairs` plots as we did above. However, when

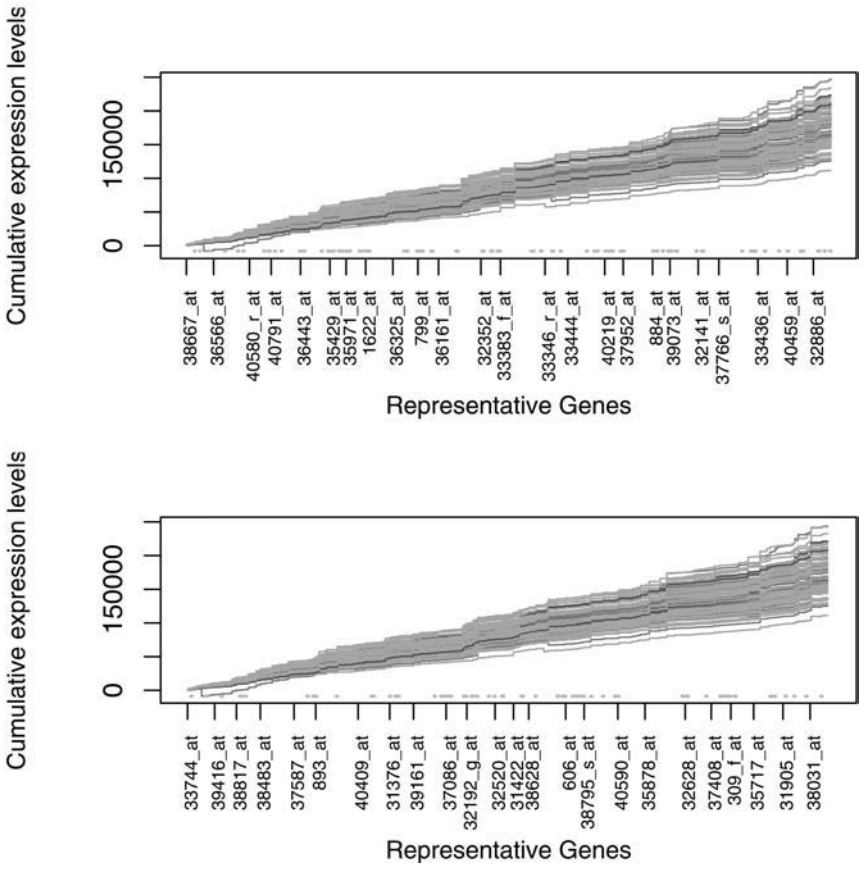

FIGURE 2.2. A plot of cumulative expression on Chromosome 17.

there are a large number of genes, this can become unwieldy. We have used the `plotcorr` function in the `ellipse` package of Murdoch (2002).

In Figures 2.4 and 2.5 we have provided these plots, one computed for each of the ERBB2-positive subset and the ERBB2-negative subset. These plots demonstrate a number of interesting features. (Note that here genes are included regardless of whether they are expressed. It might be useful to provide the expression level by coloring the ellipses.) These plots use pictographs to represent the correlation between the variables. They are symmetric.

We can see some interesting features. In both sets, we can see the clusters of highly correlated ERBB2 probe sets and TOP2A probe sets. We also see that for the ERBB2-positive patients, KIAA0130 also seems to be expressed (or at least correlated). Additionally, in this group there seems to be a high correlation between the TOP2A expression and that of SMARCE1. Further in the neighborhood of ERBB2, we can see positive correlation of ERBB2

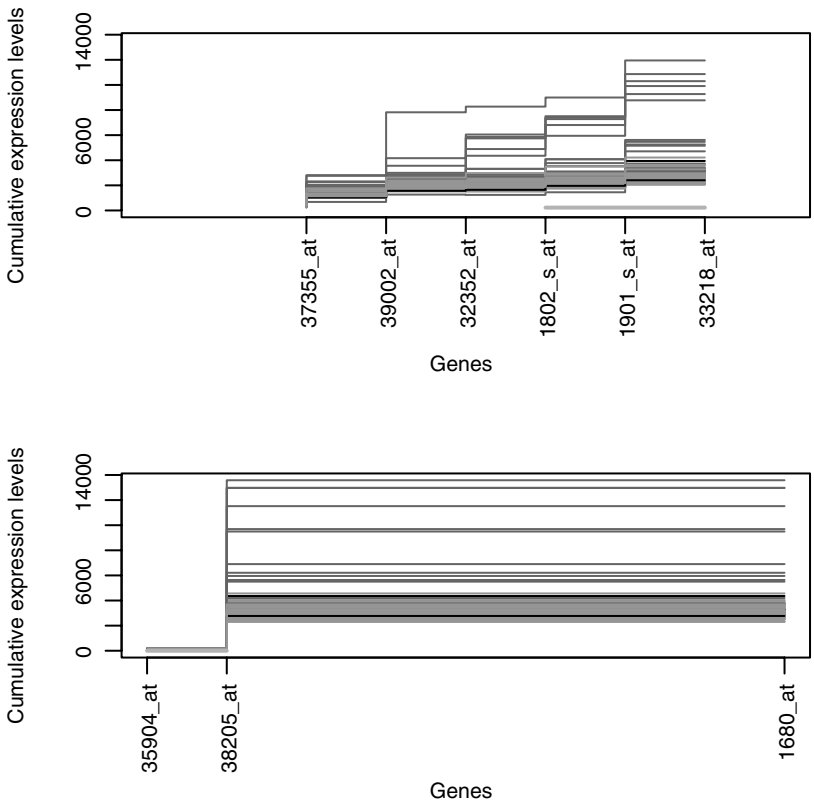

FIGURE 2.3. A plot of cumulative expression on Chromosome 17 in the region of ERBB2.

expression with GRB7 and MLN64. These relationships seem less strong in the ERBB2-negative patients.

What it means for one gene to be *near* or *close to* another gene is not yet well-understood. However, it does seem that we will want to consider concepts such as involvement in a particular pathway or process as a measure of distance as well as sharing common regulatory control as another means of finding genes that are *close to* one another. It is not entirely inconceivable that a change in regulation of one gene in these sets might be important.

Whole Genome Plots

Once chromosomal location is available for the dataset, it can be used to examine the locations of groups of genes of interest. One might be interested in whether genes group on a particular chromosome or whether they are

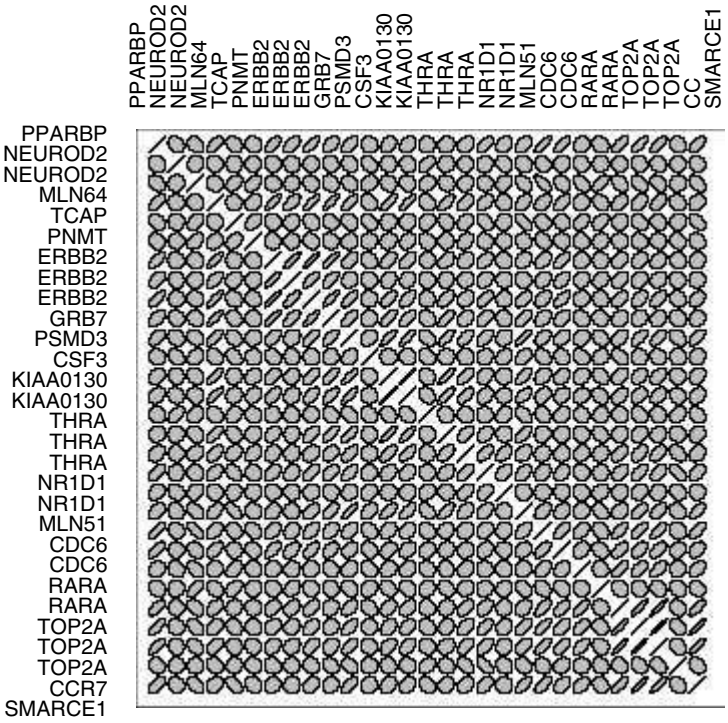

FIGURE 2.4. A plot of the correlation of expression between samples for the ERBB2-positive patients.

clustered near the ends of the chromosome or in other regions where there is some genomic instability.

We provide some tools to begin these explorations in `cPlot` and `cColor`. The first plots the location of all genes in a reference set for the genome of interest. The helper function `cColor` is then used to color locations differently.

Some form of interaction, such as brushing, would be extremely useful. We will be exploring ways of incorporating these plots into other software, such as GGobi (`www.ggobi.org`), to take advantage of the interactive facilities provided there.

Some examples of the potential use of this type of plot are:

- If gene X is of interest and we obtain data on the, say 100, ESTs with similar expression levels, then these could be colored to show where they are located.
- It is common practice to cluster genes according to expression levels subsequent to gene selection via filtering. The results of this process are

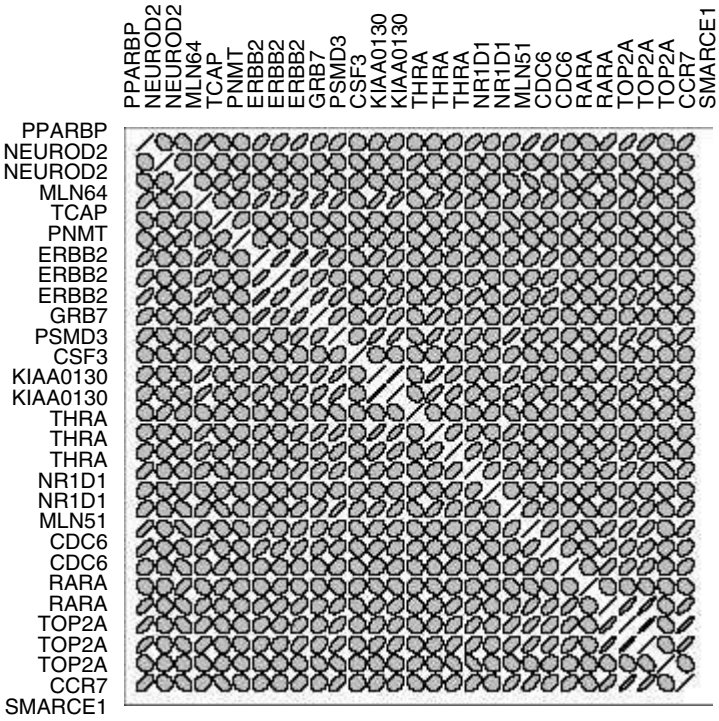

FIGURE 2.5. A plot of the correlation of expression between samples for the ERBB2-negative patients.

used to produce the usual *heat map* of genes by samples. An additional graphic showing the location of genes in different clusters may be useful.

2.6 Applications

2.6.1 A Case Study of Gene Filtering

The `golubEsets` package includes `exprSets` embodying the leukemia data reported in Golub et al. (1999). The data consist of two parts, a training set and a test set. We will show how one can perform various analyses on these data using some of the Bioconductor packages. We first load the necessary libraries and load the gene expression data objects (code not shown). For the following analyses, we will concentrate on filtering, annotation, and plotting.

The first task described in Golub et al. (1999) was the selection of the top 50 genes for predicting whether the sample was AML or ALL. We attempt a similar selection; however, rather than repeat their analysis, we will select

the 50 genes with the smallest p-value according to a t-test. Note that this is a *joint filtering* exercise—retention of a gene depends on the p-value of other genes, so the `genefilter` package is not directly applicable. First, we construct a t-test builder or constructor function. It will carry out the t-test and return the p-value.

```
ttemp <- function (m, na.rm = TRUE)
{
  function(x) {
      if (na.rm) {
          drop <- is.na(x) | is.na(m)
          x <- x[!drop]
          m <- m[!drop]
      }
      t.test(x ~ m)$p.value
  }
}
tf2 <- ttemp(golubTrain$ALL.AML)
```

At this point `tf2` is a function that will carry out a t-test. The variable `m` is bound to the value `golubTrain$ALL.AML`.

```
pvals <- esApply(golubTrain, 1, tf2)
ord <- order(pvals)
gTr50 <- golubTrain[ord[1:50],]
```

We have the top 50 genes for discriminating between these two groups using a t-test. We can find out more about these genes using the `annotate` package.

```
hg68sym<- read.annotation("hgu68sym")
hg68ll <- read.annotation("hgu68ll")
syms50 <- multiget(geneNames(gTr50), hg68sym)
ll50 <- multiget(geneNames(gTr50), hg68ll)
ll.htmlpage(ll50, "GolubTop50",
          "Top 50 ESTs discriminating ALL and AML ",
          list(syms50, round(pvals[ord[1:50]], 5)))

hg68chrom <- read.annotation("hgu68chrom")
chrom50 <- multiget(geneNames(gTr50), hg68chrom)
table(unlist(chrom50))

#output
# 1 10 11 12 13 14 15 16 17 18 19 20 3 4 5 6 7 8 9 NA
# 8  2  1  3  2  3  1  2  4  1  6  1 2 2 2 4 3 1 1  1
```

The resulting Web page can be viewed (we have it permanently on the Bioconductor Web site at `www.bioconductor.org/Docs/Papers/2002/`

Springer/GolubTop50). Note that LocusLink values and symbols that could not be resolved are reported as NA.

We can now use these 50 genes to see how well we can classify patients in the test sample.

```
gTest50 <- golubTest[geneNames(gTr50),]
gTest50  library(class)
knn1 <- knn.cv(t(exprs(gTest50)), gTest50$ALL.AML, k=3)
table(knn1, gTest50$ALL.AML)

#knn1 ALL AML
# ALL 17    0
# AML  3   14

gTest.cent <- scale(t(exprs(gTest50)), center=FALSE)
knn2 <- knn.cv(gTest.cent, gTest50$ALL.AML, k=3) #$
table(knn2, gTest50$ALL.AML)

#knn2 ALL AML  # ALL 20  0  # AML  0 14
```

Notice that centering and scaling the genes improved the prediction on the test cases. There is some reason to believe that scaling (centering makes no difference) all genes to have variance one is generally desirable. Carrying out this part of the analysis was particularly simple due to the richness of the available software for R.

Concentrating on a Single EST

We now consider a single gene in more detail and show how the use of some additional tools together with some visualization techniques provides some insight. In our analysis, reported above, one of the genes that was important in discriminating between the two classes of leukemia was the gene HOXA9, homeo box A9. Golub et al. (1999) also reported that this gene was related to clinical outcome and found that it was overexpressed in AML patients with treatment failure. The Affymetrix identifier associated with this gene is U82759_at. We can examine its pattern of expression or select other genes that have patterns of expression that are highly correlated with the pattern of expression of HOXA9.

```
gTrHox <- golubTrain["U82759_at",]
gTeHox <- golubTest["U82759_at",]
```

These data are plotted in Figure 2.6, where we can see that there is some indication that HOXA9 has a higher level of expression in the samples from AML patients than in the ALL patients for both the test and the training samples.

The function genefinder can be used to find ESTs that have patterns of expression that are similar to that of HOXA9.

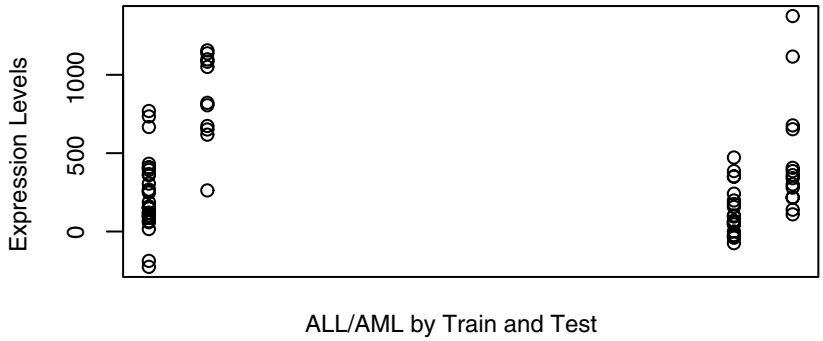

FIGURE 2.6. A pairs plot of the expression data for probes for the gene ERBB2.

```
gStr <- genefinder(golubTrain, "U82759_at", num=100)
gSte <- genefinder(golubTest, "U82759_at", num=100)

sum(gSte[[1]]$indices %in% gStr[[1]]$indices)
# 20

##could just look at AML
gTrAML <- golubTrain[,golubTrain$ALL.AML=="AML"] #$
gTeAML <- golubTest[,golubTest$ALL.AML=="AML"] #$
gStrAML <- genefinder(exprs(gTrAML), "U82759_at", num=100)
gSteAML <- genefinder(exprs(gTeAML), "U82759_at", num=100)

##now how many in common?
sum(gStrAML[[1]]$indices %in% gSteAML[[1]]$indices)
#9
##what if we compare to gSte
sum(gSteAML[[1]]$indices %in% gSte[[1]]$indices)
#33
```

2.6.2 Application of Expression Density Diagnostics

In this example, we are concerned with understanding the diversity of expression distributions presented in two strata of 89 women with breast cancer. The women were clinically classified into 47 cases of metastasis (lymph node positive) and 42 controls (lymph node negative). Although our ultimate aim is to identify features of gene expression that are predictive of avoidance of metastasis, in this example we are concerned with the diversity of gene-specific distributions within strata.

The 89 U95v2 array results are stored in the `exprSet BCes`. This was filtered to eliminate those genes for which at least 23 of the samples had a measurement above 100. There were 7297 genes remaining:

```
load("BCes.rda")

kF <- kOverA(23, 100)
ff <- filterfun(kF)
wh <- genefilter(exprs(BCes), ff)

nGenes <- sum(wh)
BCok <- BCes[wh,]
```

We can view a summary of the `BCok` dataset.

```
Expression Set (exprSet) with
        7297 genes
          89 samples
               phenoData object with 3 variables and 89 cases
            varLabels
       Chip: chip number
       lymph.nodes: indicates whether metastasis was detected
                    in the lymph nodes
       HER2: her2 status, p-positive, lp-low positive,
                                      n-negative
```

Two `exprSet`s were then extracted corresponding to cases and controls.

```
CasES <- BCok[ ,BCok$lymph=="positive"]
ConES <- BCok[ ,BCok$lymph=="negative"]
```

The function `edd.unsupervised` was applied to each set, using the `"nnet"` shape-matching method. An example call for the metastatic samples is

```
library(edd)
set.seed(12345)
# needed because nnet has random initialization
CasNN <- edd.unsupervised(CasES, "nnet")
ConNN <- edd.unsupervised(ConES, "nnet")
```

Running with both metastatic and nonmetastatic samples, we find

```
  table(CasNN)
CasNN
 b28 b82 csq1   ln mix1 mix2  n01   t3   u
2682  70   55 1059  396   38 1422 1421 154
  table(ConNN)
ConNN
 b28 b82 csq1   ln mix1 mix2  n01   t3   u
2633 140  142 1505  398   33 1109 1155 182
```

Here, the reference catalog of distributional shapes consists of $\beta(2,8)$, $\beta(8,2)$, χ^2_1, standard lognormal, $.75N(0,1) + .25N(4,1)$ (`mix1`), $.25N(0,1) + .75N(4,1)$ (`mix2`), $N(0,1)$, t_3, and $U(0,1)$. We see that the majority of genes in both cases and controls are found to have a shape resembling $\beta(2,8)$, which is a right-skewed distribution with compact support. The prevalence of right-skewness (present in the $\beta(2,8)$ along with the χ^2_1 and lognormal shapes) is consistent with the prevailing tendency to apply log transformation. However, we note that there are nontrivial numbers of genes with distributional shape matching $N(0,1)$, thus not requiring transformation, and genes with distributions shaped like clearly multimodal mixtures. The joint classification of gene-specific distributional shapes can be tabulated easily:

```
  table(CasNN,ConNN)
      ConNN
CasNN   b28 b82 csq1  ln mix1 mix2 n01  t3  u
  b28  1127  26   42 650  175    5 275 316 66
  b82    15   7    0   2    3    0  25  13  5
  csq1    6   0   22  23    2    0   0   2  0
  ln    381   3   44 425   50    0  53  94  9
  mix1  158   5    9  75   24    2  54  59 10
  mix2    7   5    0   3    0    1  14   8  0
  n01   462  53   10 103   74   14 342 315 49
  t3    417  36   14 217   61    9 311 321 35
  u      60   5    1   7    9    2  35  27  8
```

We note from this table that there are 53 genes for which the controls present an expression distribution having the shape of a standard Gaussian, while the cases present expression distribution with lognormal shape. Additionally, from the summary above, there are 434 genes for which the cases appear to have a mixture distribution.

Figure 2.7 gives substance to these distinctions by providing gene-specific density estimates for all genes in these strata defined by distributional shape pattern matching. All gene distributions were transformed on a gene-specific, stratum-specific basis to median zero and unit MAD prior to density estimation. The top panel is the set of density estimates for those genes classified as shape `mix1` among cases. There is a clear tendency to present a second mode at about $x = 2$ MADs from the median. Genes for which the stratum-specific distribution appears to be multimodal may have particular interest in that they may help unearth new diagnostic categories or may be the basis for screening procedures (Pepe et al., 2001).

The middle and bottom panels of Figure 2.7 contain density estimates for the 34 genes on which controls were found to have approximately Gaussian distributions; while cases were found to have approximately lognormal shape. In general, simple t-tests to contrast case and control expression distributions in this group of genes will be suboptimal.

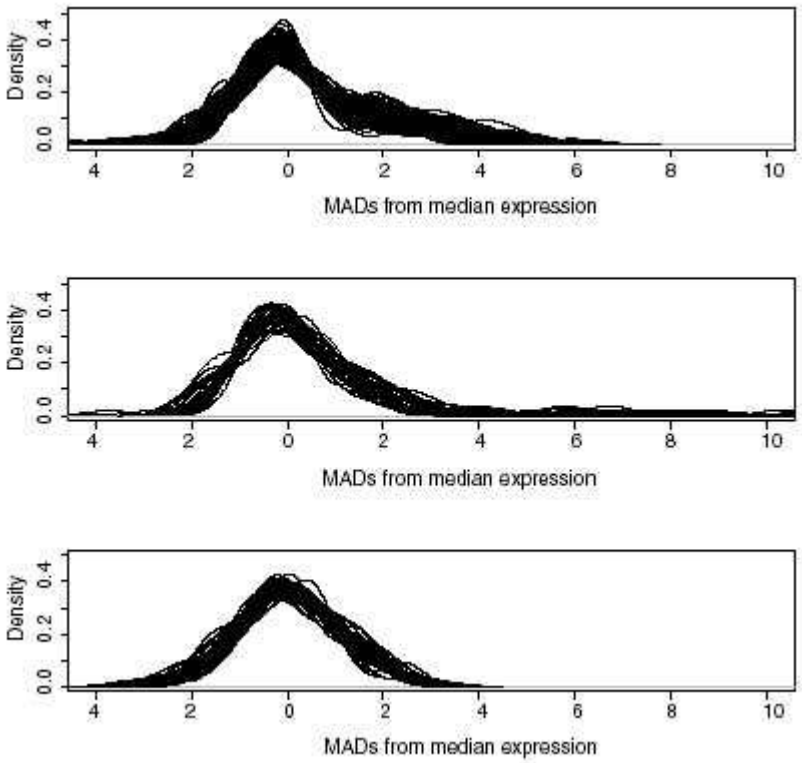

FIGURE 2.7. Superimposed default kernel density estimates for all genes possessing common distributional shape according to the `edd` procedure. Top panel: 396 genes from metastatic patients classified as `mix1`. There were 53 genes for which cases were found to have lognormal shape and controls were found to have Gaussian shape. The middle (resp. bottom) panel provides density estimates for these genes as measured on cases (resp. controls).

This application of expression density diagnostics is primarily conceptual. It exposes the existence and form of diversity of expression distributions within and across strata and genes. Work is in progress on using the results of expression diversity studies to guide the choice of discriminatory tests to increase the power of gene discovery exercises.

2.7 Conclusions

The Bioconductor project endeavors to enrich R, an interactive statistical computing and graphics environment, so that it may serve as a resource of very broad utility to bioinformatics. In this chapter, we have focused on

three processes in the analysis of expression array data: filtering from very large collections of genes to more manageable sets and then annotating and visualizing the resultant data.

Filtering is supported by the adoption of a simple and extensible data structure for the collection of multiple microarrays and associated phenotypic data and by the creation of a highly flexible interface that permits concise programming of arbitrarily complex statistical predicates to define filter behavior. A tool (expression density diagnostics) that helps guide the appropriate choice of filter by characterizing gene-expression distributions has also been reviewed.

Annotation is supported in two independent ways. First, images of biological annotation data have been introduced into R to permit use of various vocabularies to identify and interpret genomic phenomena. These images may be navigated interactively or programmatically and are conveniently updated as required. Second, functions that mine the World Wide Web interactively for up-to-the-minute interpretation of detailed analytical findings have been introduced to R. Several graphical tools that help add pictorial guidance to experiment interpretation have been discussed. These graphical modules interoperate with the annotation infrastructure so that pictures are as biologically informative as possible.

Efficient progress in bioinformatics requires that barriers to entry be lowered for both statisticians and biologists. Statisticians must be able to move conveniently, with clear documentation and a rich stock of examples, into the vocabulary and experimental frameworks of high-throughput genomics. Biologists must be able to understand and apply statistically sound protocols for experimental design and inference. Again, documentation and examples are a central resource for supporting such cross-disciplinary connections. Our objective in the Bioconductor project is to provide an open-source, integrated platform that is useful to both statisticians and biologists. The efforts of both the Bioconductor and R core developer groups are gratefully acknowledged as central to the pursuit of this objective.

Acknowledgments. Robert Gentleman's work is supported by NIH/NCI Grant 2P30 CA06516-38, by the Dana-Farber/Harvard SPORE in Breast Cancer from the NCI and by the High Tech Industry Multidisciplinary Research Fund at the Dana-Farber Cancer Institute. Vincent Carey's work was supported in part by U.S. NHLBI Grant HL66795, Innate Immunity in Heart, Lung, and Blood Disease.

We would like to thank the members of R core and Bioconductor core for providing help, discussion, and interesting, useful code. We would like to thank Drs. A. Richardson, J. D. Iglehart, and S. Chiaretti for helpful discussions on the biological aspects of modeling genomic data.

References

Brazma A, Hingamp P, Quackenbush J (2001). Minimum information about a microarray experiment: Towards standards for microarray data. *Nature Genetics*, 29:365–371.

Chambers JM (1998). *Programming with Data: A Guide to the S Language*. Springer-Verlag New York.

Eisen MB, Spellman PT, Brownand PO, Botstein D (1998). Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868.

Gentleman R, Ihaka R (2000). Lexical scope and statistical computing. *Journal of Computational and Graphical Statistics*, 9:491–508.

Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh M, Downing JR, Caligiuri MA, Bloomfield CD, Lander ES (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. *Science*, 286:531–537.

Helmreich, EJM (2001). *The Biochemistry of Cell Signalling*. Oxford University Press: Oxford.

Murdoch D (2002). *ellipse: Functions for drawing ellipses and ellipse-like confidence regions*. `http://cran.r-project.org/`.

Pepe MS, Longton G, Anderson G, Schummer M (2001). Selecting differentially expressed genes from microarray experiments. Technical report, University of Washington: Seattle.

Shalon D, Smith SJ, Brown PO (1996). A DNA microarray system for analyzing complex DNA samples using two-color fluorescent probe hybridization. *Genome Research*, 6(7):639–645.

Zhang X, Derdeyn C, Gentleman R, Leykin I, Monti S, Ramaswamy S, Wong WH, Golub TR, Iglehart JD, Richardson AL (2002). Molecular determinants of lymph node metastasis in breast cancer. Submitted for Publication.