BIOINFORMATICS

Vol. 20 no. 3 2004, pages 307–315
DOI: 10.1093/bioinformatics/btg405

affy—analysis of Affymetrix GeneChip *data at the probe level*

Laurent Gautier[1,], Leslie Cope[2], Benjamin M. Bolstad[3] and Rafael A. Irizarry[4]*

[1]Center for Biological Sequence Analysis (CBS), Technical University of Denmark, Building 208, 2800 Lyngby, Denmark, [2]Department of Mathematical Sciences, Johns Hopkins University, 3400 N. Charles St, Baltimore, MD 21218, USA, [3]Group in Biostatistics, University of California, Berkeley, CA 94720, USA and [4]Department of Biostatistics, Johns Hopkins University, 615 N. Wolfe St E3035, Baltimore, MD 21218, USA

Received on April 21, 2003; revised on July 4, 2003; accepted on July 11, 2003

ABSTRACT

Motivation: The processing of the *Affymetrix GeneChip* data has been a recent focus for data analysts. Alternatives to the original procedure have been proposed and some of these new methods are widely used.

Results: The *affy* package is an **R** package of functions and classes for the analysis of oligonucleotide arrays manufactured by *Affymetrix*. The package is currently in its second release, *affy* provides the user with extreme flexibility when carrying out an analysis and make it possible to access and manipulate probe intensity data. In this paper, we present the main classes and functions in the package and demonstrate how they can be used to process probe-level data. We also demonstrate the importance of probe-level analysis when using the *Affymetrix GeneChip* platform.

Contact: laurent@cbs.dtu.dk

INTRODUCTION

Expression microarrays are now standard tools in genetic research. It is easy to forget that just a few years ago microarray technology was a cottage industry. Chips were custom made in individual labs by the very researchers who used them and analyzed the results. It is now possible to purchase high quality microarray chips ready-made, complete with a suite of equipment and proprietary analytic software. One can carry out the entire process without considering the analytic procedure for converting a scanner image into final measures of expression. Statisticians naturally believe that it is always important to consider the analytic process. This is especially important when working with microarrays because the technology is still very immature. Analytic procedures for microarray expression data even lag behind the physical technology and users should proceed with eyes wide open.

*To whom correspondence should be addressed.

Affymetrix oligonucleotide chips (Lockhart *et al.*, 1996) are a primary example of the commercial microarray product. Affymetrix GeneChip use a set of 11–20, oligonucleotide probes, each 25 bases long, to represent a gene. The perfect match (pm) probe is designed to hybridize only with transcripts from the intended gene. In most arrays, Affymetrix pairs each pm probe with a mismatch (mm) probe, designed to measure non-specific hybridization. The mm probe differs from the pm only in the 13th base. The expression level for a gene is a summary of the data from the entire probe set. The manufacturer provides analytic software (Affymetrix, 1999, 2002) requiring very little input from the user. GeneChip enjoys widespread use, and because of this, alternative probe set summary methods have been implemented by outside organizations. Two examples are *dChip* (Li and Wong, 2001a) and *GAPS* (Selinger *et al.*, 2000).

These tools have been well received and widely used. The simple availability of high quality alternatives is beneficial to the researcher who can now often find an off-the-shelf product to meet her needs. Still further benefit can be obtained with the open source implementation of such tools in a common scripting language.

In this paper, we present *affy*, a package of functions for the storage, management and analysis of *Affymetrix* probe level data. The *affy* package is written in the open source, statistical scripting language **R**, and released under the GPL license to guarantee the continuing availability of the source code. Complete, ready to use analytic routines are available in the package, including implementations of some of the most popular and successful algorithms. With a young technology however, real strength lies in flexibility. The extensive library of statistical routines built into **R** makes is easy to customize or extend existing functions. Also *affy* is integrated into the Bioconductor project (http://www.bioconductor.org/), a collaborative effort to provide a single flexible environment

for the management and analysis of data from any microarray technology.

Section 1 describes the general design of the package. In Section 2, some of the main features are summarized. Section 3 demonstrates some of the benefits of careful attention to probe level data, and to the details of analysis.

SYSTEMS AND METHODS

We chose to implement the *affy* package in the **R** statistical software program and scripting language for several reasons. **R** is an open source program, freely available for most common computing platforms. There is an extensive built-in library of mathematical, statistical and graphical functions, with many more available as add-on packages. The object-oriented programming approach (Chambers, 1998) offers an intuitive working environment. In sum, **R** makes it is quite easy to store, manipulate and analyze data, and this is exactly what we want. We believe that the researcher using microarrays should be actively involved in analysis, and the primary goal of the *affy* package is to make this easy to do.

A few words about object-oriented programming will make the remainder of this section easier to follow. More in-depth discussions can be found in references like Nerson (1992) and Nierstrasz (1989). A class is the description or definition of objects of a particular type. It is a data structure consisting of *attributes* and *methods*, i.e. a set of variables and functions, belonging to objects of that type. An *instance* of a class is a specific object of that type. Class-specific definitions of generic functions, like `plot` and `print`, allow a user to access complex procedures for specific data structures using uniform calls. Additional comments about the integration of the *Affymetrix* design in a **R** package can be found in Irizarry *et al.* (2003).

Data structures

Consider a typical GeneChip microarray experiment. A total of m samples are hybridized to $n \geq m$ chips, according to a predetermined experimental design. All n chips are of the same type and a common Chip Description File (CDF file), summarizes identifying information for each probe cell on the chip. After each chip is scanned and the image analyzed, probe intensity data for the chip is recorded in a CEL file. Unless one wishes to revisit pixel-level image analysis, n CEL files and 1 CDF file contain all data necessary for further work. The *affy* package is not intended to be imaging software, therefore it does not specifically accommodate pixel-level data. The CDF and CEL files are therefore our natural starting point.

The most fundamental data structures in the package are defined in the classes *Cdf* and *Cel*, corresponding to the files of the same types. Like CEL files, the *Cel* data structure is very simple, primarily storing all the probe intensities for a single chip. Information about probe identity, the location of each probe on the chip, and very limited sequence data is stored in a *Cdf* object. An additional structure called a *Cdf environment* is also defined. Using hash tables, this environment allows efficient mapping of probe set identifiers to probe indexes. Using these indexes, *Cel* the intensity for any named probe can be extracted and associated with the corresponding *Cdf* information. A large number of *Cdf environments* are provided for download, and facilities are provided to build *Cdf environments* from CDF files for which they are not currently available.

The *AffyBatch* class is designed to organize the data from a single microarray experiment. This data structure is foremost a container for several *Cel* objects, but also includes forms for extensive documentation of an experiment. One of its attributes is of class *MIAME*, allowing a user to enter and retrieve documentation rapidly in accordance with the MIAME standard (Brazma *et al.*, 2001). The *ProbeSet* class accommodates probe intensity data for a single probe set. Table 1 summarizes the main classes and structures.

Table 1. Brief textual description of the classes and data structures

Cdf	This structure stores probe identity data read from a *Chip Definition File*. Cdf data is not chip-specific, but is common to all chips of a given type
Cdf environment	This associative data structure is used to map probe identifying information to the corresponding probe intensities
Cel	This structure stores probe intensity data from a single chip. Data is read from the *CEL* data file corresponding to that chip
AffyBatch	This structure groups `Cel` data from a set of chips with common `Cdf` into a single structure. The object also includes variables for experiment documentation
ProbeSet	This structure contains the signal intensity data for a single probe set across several chips

Functional structures

Even the best data storage structure is useless without functions to enter, retrieve and process the data stored there. We have implemented class-specific versions of several standard **R** functions. Generic commands for plotting or summarizing data, including `plot(x)`, `boxplot(x)`, `hist(x)`, `summary(x)`, `x` do different things, depending on the class of the object `x`. Additional built-in functions offer quick, intuitive access to subsets of a Cel or an AffyBatch object. Thus `pm(x)` and `mm(x)` retrieve pm and mm probe data, respectively, while the command `geneNames(x)` returns a list of probe set identifiers. See package documentation for a full list of accessor functions.

When designing the classes, their respective attributes and methods, particular care was taken to balance efficiency and flexibility for the most common Use Case: the processing of probe level data into gene expression measures. As presented in Figure 1 we distinguish four separate processing

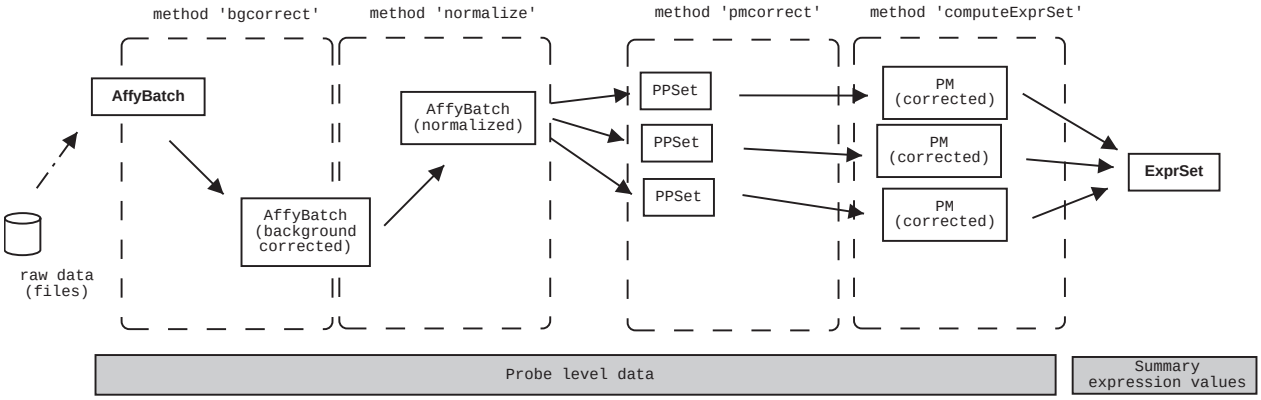

Fig. 1. A common Use Case is to go from the raw data to expression summary values. Here, we start with an *AffyBatch* object. The call to the method *normalize* performs the normalizing transformation of the probe level data. The call to the method *computeExprSet* performs the summary of the probe-level intensities in a probe set into one expression value. Background correction and normalization typically use chip-wide features of the data, while the pm correction and probe set summary steps usually use features specific to a probe set.

steps: background correction, normalization, pm correction and summary expression value computation. A detailed description of these steps and of the algorithms currently implemented for each is given in the next section.

The user picks one algorithm for each step, and a single function `expresso` runs them in sequence. Affymetrix's current algorithm, *MAS 5.0*, their original method *MAS 4.0*, and Li and Wong's *dChip* are among the methods implemented in the package. It should be noted that the implementations of these methods included in the package may differ slightly from the release versions available from the authors. The versions included in the package were prepared from published descriptions of the algorithm, and without access to final, low-level programming decisions. In at least one case, an attempt was made to quantify the differences. The results are available on the webpage http://www.stat.berkeley.edu/bolstad/MAS5diff/Mas5difference.html. The modular design makes it easy to mix and match algorithms as well. For example, one could write her own routine to correct for background noise in the image, skip the normalization procedure, and then proceed according to Affymetrix's MAS 5.0 algorithm.

Several convenience features are included in the package as well. Although these features are not fully implemented in the current release, there are forms for convenient entry of some documenting data, and graphical user interface access to some of the analytic functions. Those who wish to use one of the popular expression measures in its entirety and with push-button convenience will find simple commands to do this.

ALGORITHMS AND IMPLEMENTATION

In this section we describe some of the features of the package in detail. In particular, we will focus on methods for acquiring data from CEL and CDF files and on algorithms for processing probe intensities into expression measures.

Entering data

It is very easy to read CEL file data into Cel and AffyBatch objects. The functions `read.celfile` and `read.affybatch`, included in the package, do this. MIAME experiment description data, information on experimental design, and supplemental covariate information can be read in from formatted files at the same time. Those who prefer a graphical user interface for data entry can use the function `ReadAffy` in place of `read.affybatch`. The functions `merge.AffyBatch` and `split.AffyBatch` facilitate re-organization of experimental data once it is in the *R* environment.

Cdf environments for the most widely used GeneChip are available from *Bioconductor* as add-on packages so there is often no need to read in Cdf data. Those using custom chips, or chips for which Cdf packages are not yet available can import CDF file data using the `read.cdffile` command. *Bioconductor* includes facilities for building new Cdf environments as well.

It is important to note that the class structure is independent of the file formats. Thus, if *Affymetrix* makes changes to the file format, only the reading functions will change. The class structure will be preserved.

Processing steps

There are several built-in methods for each step in the process of turning probe intensity data into expression measures. Individual users can easily modify these, or add their own. In the following paragraphs, we review the currently available methods.

Background correction There is some amount of background noise in every scanner image. Sterile water can be

labeled and hybridized to a microarray. Even though there is then no RNA in the sample, the scanner will detect low levels of fluorescence on the chip. In both *MAS* algorithms, the distribution of probe intensities is used to estimate overall background noise level and adjust for it. Another method of estimating background is the convolution of signal and noise distributions used by the RMA method (Irizarry *et al.*, 2003).

Normalization No step in the hybridization process can be perfectly controlled. The quantity of RNA in a sample varies slightly from chip to chip. Even if the exact same sample is used on each of several chips, there will be chip to chip differences in the overall distribution of probe intensity values. Normalization procedures attempt to detect and correct systematic differences between chips so that data from different chips can be directly compared. Studies show that the normalizing procedure has a marked impact on the final expression measures (Bolstad *et al.*, 2002). A number of normalization procedures for Affymetrix GeneChip have been proposed and implemented and several are included in the *affy* package.

Details concerning the normalization methods included in the package can be found in their respective references: `loess` (Åstrand, 2003), `invariantset` (Li and Wong, 2001b), `qspline` (Workman *et al.*, 2002) and `quantiles` (Bolstad, 2001). Since the current release of the package, Huber *et al.* (2002) have added an additional normalization method, and developed *R* code to aid in developing normalization methods. While the code remains in an external package, the functions are easily integrated with those in the *affy* package through its extension capabilities.

pm correction Mismatch probes are included on *Affymetrix GeneChips* to quantify non-specific and cross-hybridization. Originally, mm signal was subtracted from the pm signal to correct for non-specific and cross hybridization (Affymetrix, 1999). In the current release of the software (Affymetrix, 2002), *Affymetrix* uses a different approach in which an idealized version of the mm signal is subtracted from each mm probe. Many researchers prefer to ignore the mm probes entirely and use uncorrected pm probes alone. Others use detailed probe sequence information to develop sophisticated correction methods. At this time only the *MAS 4.0* and *MAS 5.0* pm correction methods are implemented.

Computation of expression values from probe intensities Each gene is represented on the GeneChip by one or more probe sets. Each probe set includes 11–20 probe pairs. We think of an expression value for a gene as a summary of the corresponding probe-level data. Several researchers have developed their own summaries of Affymetrix probe-level data, and this is an area of continuing research. A number of summary methods are implemented in the current release of the *affy* package. These include the simple trimmed average of *MAS 4.0* and the more robust procedure of *MAS 5.0*, based on Tukey's biweight. In addition, the model-based method implemented in dChip by Li and Wong (2001a), the non-parametric model suggested by Lazaridis *et al.* (2002) and a method fitting an additive model using Tukey's median polish procedure are implemented.

DISCUSSION

The benefits of probe level analysis

The *affy* package is designed to balance user control of data analysis with convenience. Graphical user interfaces, object oriented programming and modular function design enhance the convenience of the package. Nonetheless, the balance is skewed greatly in favor of user control. We believe that this is appropriate. Microarray technology is still quite new and although the benefits of easy to use, commercial chips cannot be overestimated, the cautious user will retain as much hands on control of the process as possible.

We firmly believe that the open source philosophy is compatible with the spirit of scientific research. Both **R** and the *Bioconductor* project are open source efforts, encouraging contributions from independent researchers. In this spirit, we offer the *affy* package not as a finished product, but as a beginning. In the following paragraphs, we describe some of the benefits we see in the approach taken here. Some of the discussion will focus on features already implemented in the package. Other portions describe areas of future research, and functionality that could be added by any interested party. We used a publicly available data set of reasonably large size to illustrate the discussion: the 102 chips of type *HG-U95Av2* from Singh *et al.* (2002). We refer to it as the *prostate tumor* data set below.

The package includes implementations of the most popular expression measures for *Affymetrix* GeneChip. Modular function design allows the user to compare easily competing methods and mix and match the best features of each. Another advantage of modular design is the ability to isolate a single step in the process and examine it in detail.

The *affy* package includes a suite of quality control checks. Hybridization is a complicated chemical process and there are several points at which things can go wrong.

Histograms and boxplots of PM probe data [`hist(x)` and `boxplot(x)`] offer a picture of the overall distribution of probe intensities found on each array. One or more arrays with probe intensity distributions very different from the others in an experiment may be flawed. Histograms showing a great deal of mass at high intensity values could indicate a saturation problem. RNA molecules are unstable and subject to degradation characteristically starting from the 5′ end of each transcript. Individual probes in each probe set are numbered starting from the 5′ end of the transcript, so relative position within the transcript is known. The RNA degradation plot [`plotAffyRNAdeg(x)`] shows mean expression as a function of relative position to detect poor quality RNA. A routine visual inspection of the scanner image from each

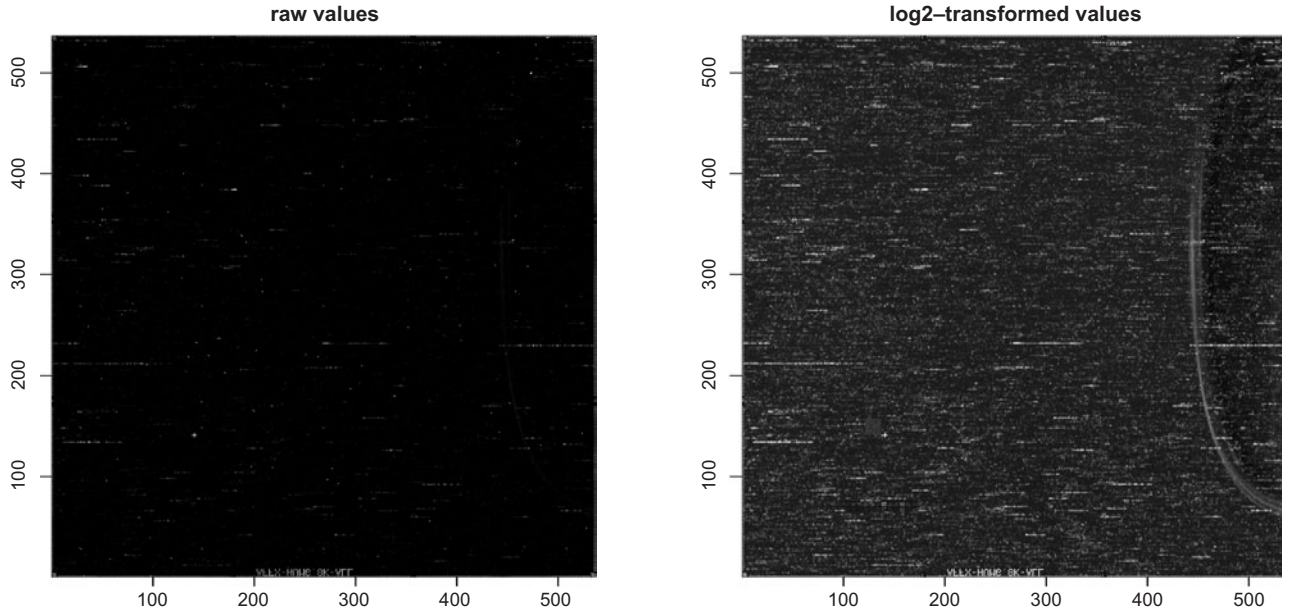

Fig. 2. A reconstruction of the original scanner image, perhaps augmented by an appropriate numerical transformation of the data, may reveal defects on a chip. Here, we see an air bubble or a scratch [data from Workman *et al.* (2002)].

chip, in order to detect obvious experimental artifacts such as air bubbles or salt stains caused by a problem during one of the washing steps, is a recommended quality control step. The `image(x)` function reconstructs the scanner image from Cel data (Fig. 2). Notice that 'zooming-in' Figure 2 would make clearer the existence of an artifact. With access to the probe-level data this is relatively simple to do.

It is easy to manipulate and visualize probe level data using the *affy* package and custom quality control procedures can easily be implemented. On each GeneChip there are several probe sets corresponding to controls. These are easily identified by the letters *AFFX* at the beginning of the probe identifier. Some of them are *spiked-in* control genes. The matching targets are added in precise quantities and at different steps in the labeling and hybridization process. These transcripts have predictable expression levels and so the probe sets play an important quality control role in the *MAS* algorithm. Several independent researchers have successfully implemented their own spike-in protocols in conjunction with specialized quality control tests. Because input is controlled, outcomes are predictable, and it is possible to discover and correct sources of error. Using the *affy* package, it is straightforward to isolate and examine the data from these probe sets, as shown in Figures 3 and 4.

Data visualization is important for expression level data, both for quality control, and to evaluate competing expression measures. As a example, we applied two different expression methods to the prostate tumor data set (Fig. 5). In this particular case, prior biological knowledge leads us to expect that the vast majority of genes will be similarly expressed in normal and tumor samples. Thus we prefer a measure that results in a single common distribution of expression values over one that shows different distributions for tumor and normal samples.

Finally, we present an example that demonstrates the flexibility one has using the package. One of the experimental protocols described by *Affymetrix* consists of adding $50\,pM$ of an internal control called *biotinylated B2*. This control sequence hybridizes to probes located all along the edges of the chip, delimiting the outer boundary.

The biotinylated B2 probes are among a large group that do not have probe set identifiers in the CDF file. We refer to these as the *unnamed* probes. Figure 4 shows how easily one could retrieve the X and Y positions for the *unnamed* probes and plot them over the summary image of CEL data. Figure 6 shows the resulting graph which corroborates that the bounding probes are among those unnamed probes. Regularly spaced groups of probes can also be observed across the surface of the chip.

With this data available, it is now easy to perform statistical assessments using the **R** language. Figure 6 shows a histogram of the intensity values of the *unnamed* probes, which suggest there are both low- and high-intensity signal populations. Figure 6 also shows the results of a k-means cluster analysis of 6000 *unnamed* probes on a HG-U95Av2, corroborating the existence of the two groups.

Because details about these probes are not made public by *Affymetrix* we can only speculate about possible applications based on the information they provide. There are several potentially interesting applications to explore. For example, it might be possible to improve quality control by monitoring

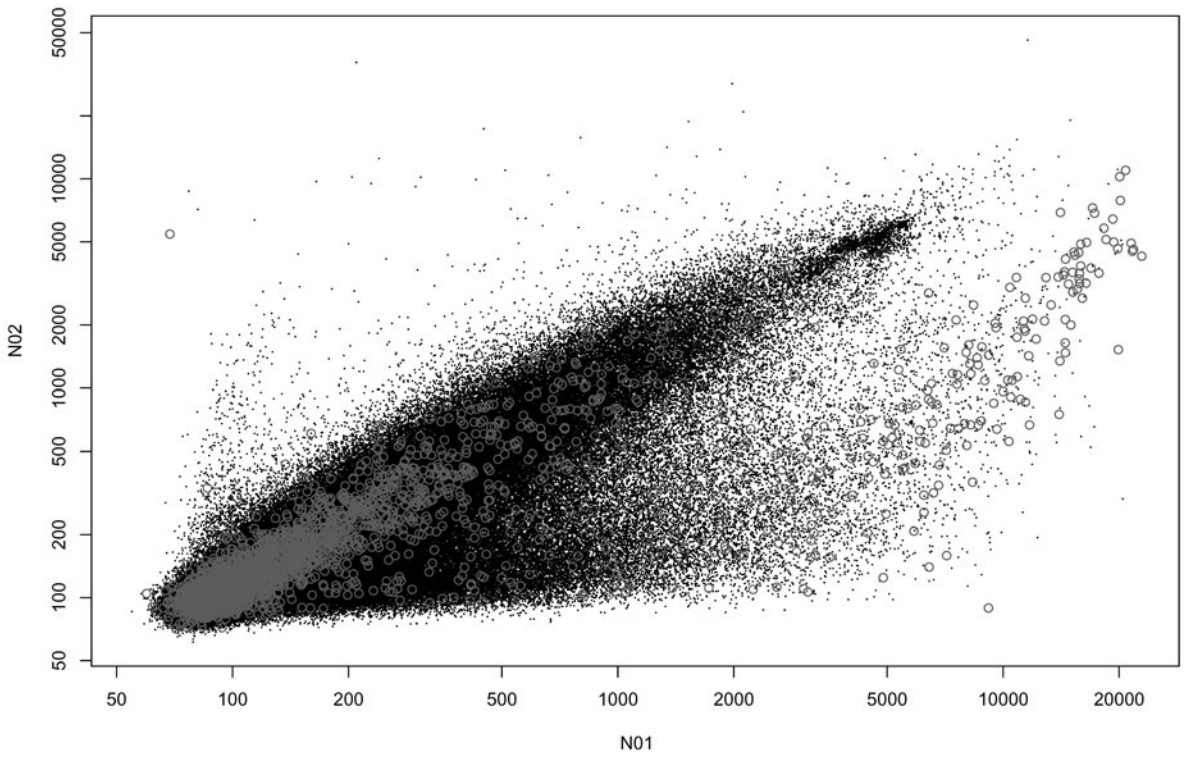

Fig. 3. Scatter plot of the probe intensities on the two first chips from the *prostate tumor* data set (*N01* and *N02* respectively). The probes corresponding to the *Affymetrix spike-in* controls (identifiers starting with *AFFX*) are plotted in grey. This figure can be viewed in colour as Supplementary Data at *Bioinformatics* online.

```
## abatch is an AffyBatch
controls.gni <- grep("AFFX.+", geneNames(abatch))
controls.names <- geneNames(abatch)[controls.gni]
controls.indices <- indexProbes(abatch, which="pm", controls.names)

## subset the probe intensities
controls.intensities <- intensity(abatch)[control.indices, ]
```

```
## import CDF data from a file
cdf <- read.cdffile("HG-U95Av2.CDF")
cel <- read.celfile("N01.CEL")

probe.type <- pmormm(cdf)
## obtain X and Y coordinates for the 'unknown probes'
unknown <- which(is.na(probe.type), arr.ind=TRUE)

image(cel)
points(unknown[, 1], unknown[,2], col="red")
```

Fig. 4. Short examples of *R*. (top) Indices for the *AFFX* controls are easily found, and the corresponding probe intensities can be isolated. (bottom) The *unknown* probes are plotted over the image of a chip.

the relative strength of the signal. Since these probes are spread all around the chip, they might also contribute to new data normalization procedures. Hopefully the details will be shared with the scientific community sometime in the near future.

Integration to the Bioconductor project

The *affy* package can be used on a stand alone basis, and expression values can be saved in a tabulated text file. However it is fully integrated into the Bioconductor project, and is compatible with other analysis tools included in that project. The Bioconductor project is an open source effort toward a universal system for the storage, manipulation and analysis of microarray data of all kinds. When necessary, we have developed unique data structures suitable for GeneChip data, but when possible we have incorporated standard Bioconductor structures. Expression data structures, e.g. are not unique to the *affy* package. GeneChip data can easily be incorporated with data from other sources since there is a common environment for both. In the paragraphs that follow, we highlight a couple of the most relevant features available in the Bioconductor project.

Data sets Bioconductor offers an extensive and flexible set of structures for gene annotation. Built-in functions provide convenient access to popular databases, and Bioconductor offers complete, regularly updated annotation packages for several Affymetrix GeneChip. For each probe set on a chip the following information is accessible: ACCNUM (GenBank accession number), UNIGENE (UniGene cluster IDs), LOCUSID (Unique integer ID for locus), MAP

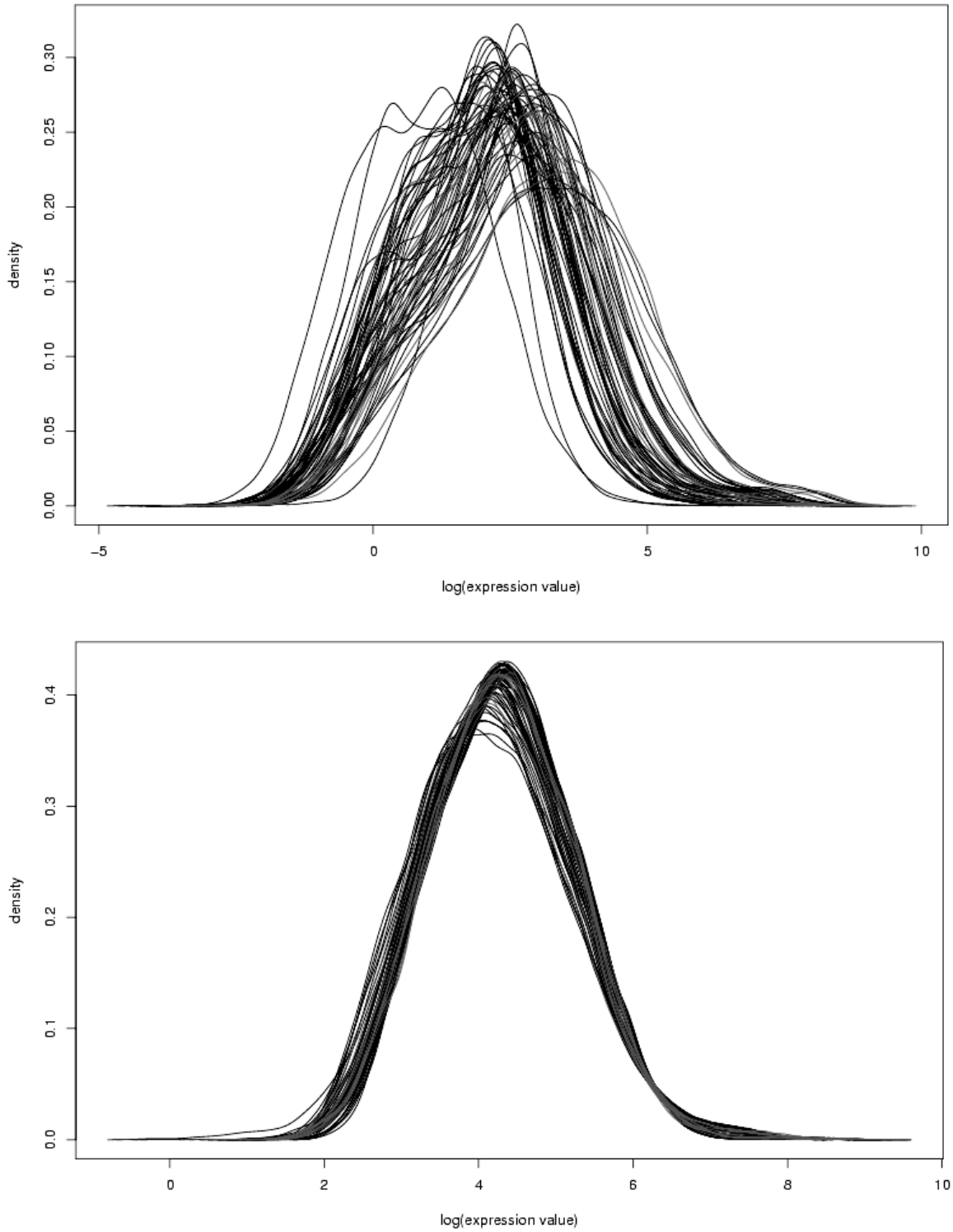

Fig. 5. Density estimates of the gene expression values for the 102 chips of the *prostate tumor* data set. The curves corresponding to normal tissues are plotted in black while the ones corresponding to tumor samples are plotted in red. (top) The expression values were obtained by subtracting an estimated background (as done in MAS 5.0), then by correcting the PM values of the probe set and computing a summary expression values as described in the MAS 5.0 white paper. (bottom) The expression values were obtained by subtracting the same estimated background as before, normalizing the probe intensities as done in the *dChip* software, using only the PM values in the probe set to compute a summary expression value as described by Li and Wong (2001a, This figure can be viewed in colour as Supplementary Data at *Bioinformatics* online).

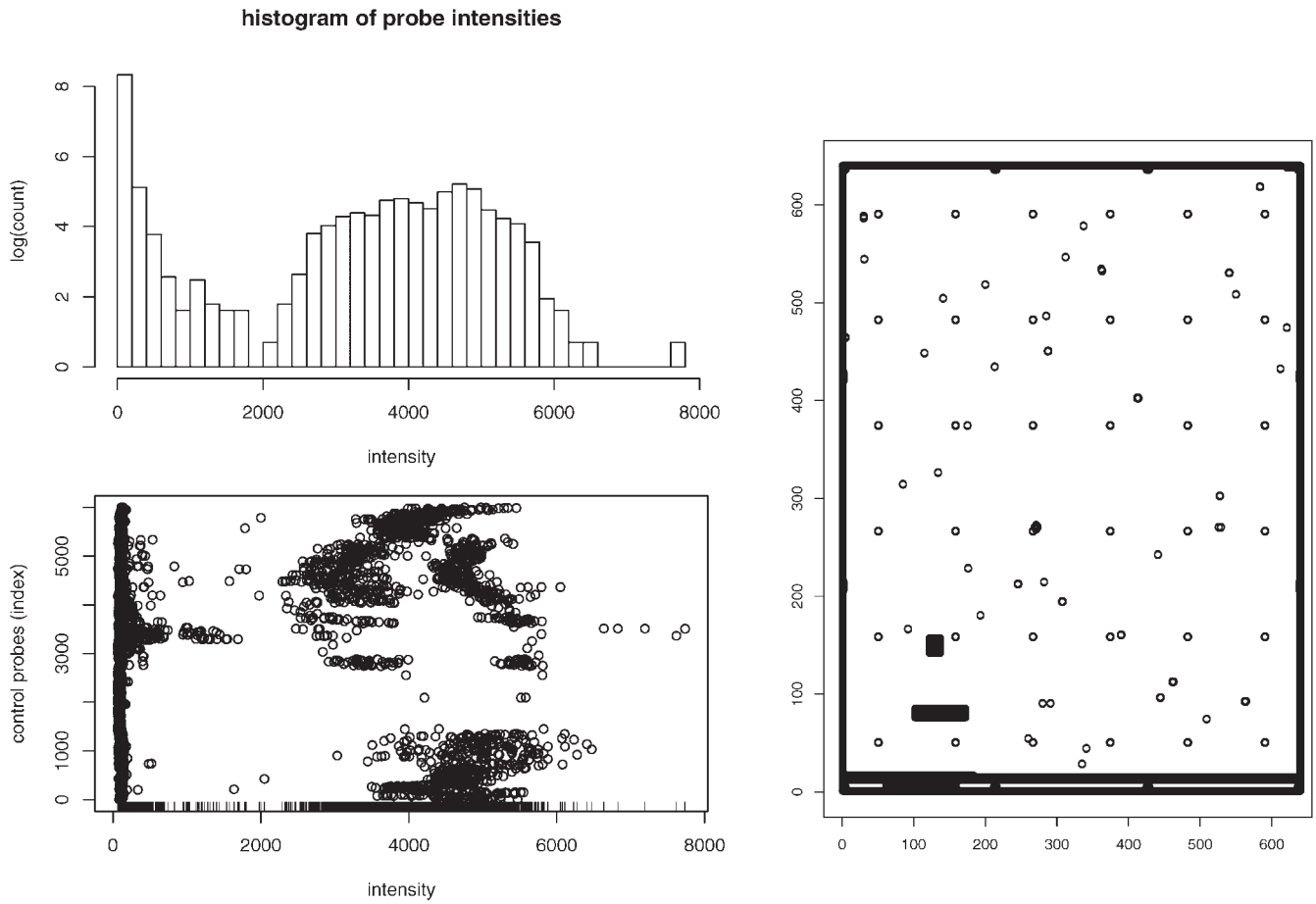

Fig. 6. Visualization of the *unlabeled* probes on a *HG-U95Av2* chip. The upper left plot shows an histogram of the probe intensities. Two populations of probes can be assumed: the first population has a low-intensity signal while the second population has a high-intensity signal. The lower left plot presents a scatter plot and a rug plot (ticks on the x-axis correspond to the x-coordinate of the points shown) of the probe intensities, colored according to which one of two classes they were assigned by k-means clustering. The right plot shows the spatial locations of these probes. The bounding probes and some of the regularly spaced groups of probes across the chip are thought to be 'biotinylated B2' probes.

(the chromosome assignment), CHR (chromosome number), PMID (a sub set of PubMed unique IDs), GRIF (PubMed unique identifier), SUMFUNC (summary of the function of genes), GO (gene ontology ID), CHRLOC (chromosomal location of genes), CHRORI (chromosomal orientation of genes), ENZYME [Enzyme Commission identifier (EC)], PATH (pathway name). A complete set of tools to build packages of annotations is available on Bioconductor and is described elsewhere (Zhang *et al.*, 2003).

tkWidgets Some graphical user interfaces are currently available in the *affy* package and others are under development. We have used the *tkWidget* package in the Bioconductor project to develop these. A combination of prepared forms and development tools make it easy to build-up interfaces for *R* functions. Researchers can use these tools to customize user interfaces for their own more convenient use, building tighter links between users and developers of the methods.

Performance

Although modularity and object-oriented design offer appreciable advantages, a price is paid in computational efficiency. However, computation of expression values seldom represents an appreciable expense. In typical cases, where an experiment includes 10–30 chips, the computation of expression values takes only several minutes on the average desktop PC.

At the extreme upper end of 32 bit architectures, the package has been used successfully on a data set of 102 *HG-U95Av2* arrays. This analysis was performed on a PC running the Linux operating system and equipped with 2 GB of memory. For larger data sets, it may be necessary to work on subsets of the data and integrate results at the end. Improvements in memory

management within the package are underway, and preliminary tests suggest that it may be soon possible to process 250–300 arrays at once.

Additionally, **R** and the *affy* package can be compiled for 64 bit architectures. If this is done, much more is possible.

ACKNOWLEDGEMENTS

The authors would like to thank Robert Gentleman for his invaluable help through comments and suggestions. Special thanks is due as well to three anonymous reviewers who's recommendations did much to improve the paper. The work of L.G. is funded by the Danish Biotechnology Instrument Center. He would also like to acknowledge the microarrays, group at CBS, among particularly Steen Knudsen (lead) and Thomas Jensen (technical expertise on the experimental side), the Bioinformatics department of the Yang-Ming National University of Taiwan and the Dana-Farber Cancer Research Center, which computing facilities were used. The work of R.A.I. and L.M.C. is in-part supported by the Hopkins PGA (www.hopkins-genomics.org) Administrative/Bioinformatics Component (P01 HL 66583).

REFERENCES

Affymetrix (1999) *Affymetrix Microarray Suite User Guide*, Version 4 edn. Affymetrix Santa Clara, CA.

Affymetrix (2002) *Affymetrix Microarray Suite User Guide*, Version 5 edn. Affymetrix Santa Clara, CA.

Åstrand,M. (2003) Contrast normalization of oligonucleotide arrays. *J. Comput. Biol.*, **10**, 95–102.

Bolstad,B.M., Irizarry,R.A., Åstrand,M. and Speed,T.P. (2002) A comparison of normalization methods for high density oligonucleotide array data based on bias and variance. *Bioinformatics*, **19**, 185–193.

Bolstad,B. (2001) Probe level quantile normalization of high density oligonucleotide arrays. submitted.

Brazma,A., Hingamp,P., Quackenbush,J., Sherlock,G., Spellman,P., Stoeckert,C., Aach,J., Ansorge,W., Ball,C.A., Causton,H.C. *et al.* (2001) Minimum information about a microarray experiment (MIAME) toward standards for microarray data. *Nat. Genet.*, **29**, 365–371.

Chambers,J.M. (1998) *Programming with Data*. Springer, New York. ISBN 0-387-98503-4.

Huber,W., von Heydebreck,A., Sueltmann,H., Poutska,A. and Vingron,M. (2002) Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, **18** (Suppl. 1) S96–S104.

Irizarry,R.A., Gautier,L. and Cope,L. (2003) *The Analysis of Gene Expression Data*, Chapter 4. Springer, Berlin.

Irizarry,R.A., Hobbs,B., Collin,F., Beazer-Barclay,Y.D., Antonellis,K.J., Scherf,U. and Speed,T.P. (2003) Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, **4**, 249–264.

Lazaridis,E.N., Sinibaldi,D., Bloom,G., Mane,S. and Jove,R. (2002) A simple method to improve probe set estimates from oligonucleotides arrays. *Math. Biosci.*, **176**, 53–58.

Li,C. and Wong,W.H. (2001a) Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection. *Proc. Natl Acad. Sci., USA*, **98**, 31–36.

Li,C. and Wong,W.H. (2001b) Model-based analysis of oligonucleotide arrays: model validation, design issues and standard error application. *Genome Biol.*, **2**, 1–11.

Lockhart,D.J., Dong,H., Byrne,M.C., Follettie,M.T., Gallo,M.V., Chee,M.S., Mittmann,M., Wang,C., Kobayashi,M., Horton,H. and Brown,E.L. (1996) Expression monitoring by hybridization to high-density oligonucleotide arrays. *Nat. Biotechnol.*, **14**, 1675–1680.

Nerson,J.M. (1992) Applying object-oriented analysis and design. *Commun. ACM*, **35**.

Nierstrasz,O. (1989) A survey of object-oriented concepts. In Kim,W. and Lochovsky,F., (ed.), *Object-Oriented Concepts, Databases and Applications*. ACM Press and Addison Wesley Reading, MA, pp. 3–21.

Selinger,D.W., Cheung,K.J., Mei,R., Johansson,E.M., Richmond,C.S., Blattner,F.R., Lockhart,D.J. and Church,G.M. (2000) RNA expression analysis using a 30 base pair resolution *Escherichia coli* genome array. *Nat. Biotechnol.*, **18**, 1262–1268.

Singh,D., Febbo,P.G., Ross,K., Jackson,D.G., Manola,J., Ladd,C., Tamayo,P., Renshaw,A.A., D'Amico,A.V., Richie,J.P. *et al.* (2002) Gene expression correlates of clinical prostate cancer behavior. *Cancer Cell.*, **1**, 203–209.

Workman,C., Jensen,L.J., Jarmer,H., Berka,R., Gautier,L., Nielsen,H.B., Saxild,H.-H., Nielsen,C., ren Brunak,S. and Knudsen,S. (2002) A new non-linear normalization method for reducing variability in dna microarray experiments. *Genome Biol.*, **3**, research 0048.1–0048.16.

Zhang,J., Carey,V. and Gentleman,R. (2003) An extensible application for assembling annotation for genomic data. *Bioinformatics*, **19**, 155–156.