CHAPTER TWO

An Introduction to Microarray Data Analysis and Visualization

Gregg B. Whitworth

Contents

Department of Biology, Grinnell College, Grinnell, Iowa, USA

Methods in Enzymology, Volume 470
ISSN 0076-6879, DOI: 10.1016/S0076-6879(10)70002-1

Abstract

Microarray experiments offer a potential wealth of information but also present a significant data analysis challenge. A typical microarray data analysis project involves many interconnected manipulations of the raw experimental values, and each stage of the analysis challenges the experimenter to make decisions regarding the proper selection and usage of a variety of statistical techniques. In this chapter, we will provide an overview of each of the major stages of a typical yeast microarray project. We will focus on providing a solid conceptual foundation to help the reader better understand each of these steps, will highlight useful software tools, and will suggest best practices where applicable.

1. Introduction

1.1. Overview

Figure 2.1 illustrates a typical data analysis scheme for a microarray experiment. The first challenge in a microarray project is to develop a clear definition of the biological question of interest, as all subsequent tasks will be guided by the goals of the study. Given the material and time costs associated with microarray experiments, it is well worth designing a data analysis scheme in tandem with your experimental approach, to ensure that the data you produce will be sufficient to rigorously address your question of interest. Once an overall goal for the project is set, the key experimental design choices include the array platform, probe construction and sequence, and experimental protocol. As these aspects of a microarray experiment are covered elsewhere in this series,[1] below we will focus our discussion on the experimental design decision which has the greatest impact on data analysis: single-sample versus competitive hybridization. Once an experiment has been performed, data analysis begins with the acquisition of digital images describing the signal intensity associated with each "spot" or probe on an array. Image analysis results in a table linking each probe on the array with a quantitative response value. In the preprocessing stage, these raw response values are subjected to quality control and are normalized to allow for a fair comparison of measurements between probes on a single array and among different arrays. Once a high-quality set of comparable values has been

[1] See Chapter 1 by Andrew Capaldi and Chapter 3 by Maki Inada and Jeff Pleiss in this volume.

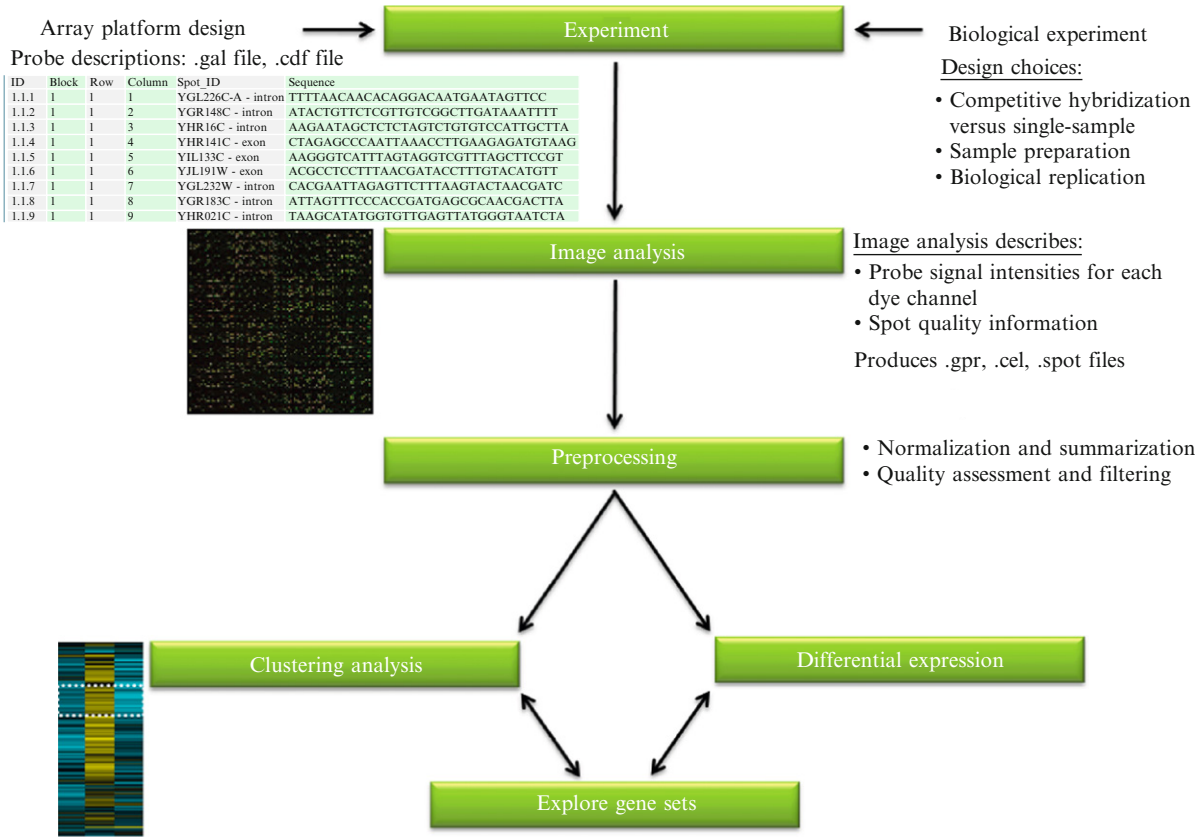

Figure 2.1 The flow of information in a typical microarray data analysis project.

obtained which describe the behavior of each unique target in each experimental sample, results can be visualized with a variety of clustering tools or subjected to statistically rigorous methods for identifying targets that are affected by each experimental treatment. Finally, after we have identified an interesting set of mRNAs, genes, or other targets, there are a number of ways to explore biologically meaningful associations among members of the set.

In this chapter, we will walk through each of these major steps in the chronology of a microarray data analysis scheme. The goals of this chapter are to introduce common terminology, provide a conceptual basis for understanding the major design concerns, and suggest useful software and statistical tools where applicable.

1.2. Commonly used terms in microarray data analysis

Microarray platforms and experimental designs have grown increasingly varied as new uses for microarray technologies have emerged. The terminology used to discuss microarray data analysis procedures is often confusing to newcomers because the vocabulary must be sufficiently abstract to describe a wide range of technologies and assays. Here are some useful working definitions:

Microarray	A microarray can be broadly defined as a high-density grid of probes attached to a two-dimensional surface.
Platform	Platform specifies a particular array technology. This includes the choice of surface material (glass, silicon), probe material (PCR product, oligonucleotide), probe targets (ORFs, SNPs, tiled genome-sequence), and manufacturing process (robotically printed, photolithography, ink-jet).
Probe	A nucleic acid bait sequence, usually a DNA oligonucleotide or PCR product, designed to hybridize to a specific target in samples loaded onto the surface of an array. Arrays may include more than one probe sequence for each target.
Spot, feature	A location on an array where a specific probe has been printed or synthesized. Many array designs contain replicate spots, meaning that a given probe is represented by more than one spot on the array.
Biological sample	The starting material from a biological experiment, such as total RNA or DNA. In some designs this is directly labeled and loaded onto the microarray, in others it is first converted (e.g., from RNA to cDNA).

Target	A molecular species in the sample for which there is a specific probe on the array.
Label	Labels are attached to targets to allow for the detection of material hybridized to each spot on the array. Usually labels are fluorescent dyes, such as Cy3 and Cy5, which are excited by a laser and detected using dye-specific filter sets.

1.3. A simple case study

Consider a simple yeast experiment which we will use to illustrate different aspects of a microarray data analysis project throughout this chapter. Suppose we are interested in performing a common class of microarray experiment: identification of genes which are differentially expressed when yeast cells are exposed an environmental stress. Our biological samples in this example might be total RNA isolated from cells before the treatment and at various time points following the treatment. Our material preparation protocol could involve a reverse transcription reaction in which we synthesize a cDNA copy of isolated RNA molecules, followed by a labeling reaction in which we attach a fluorescent dye to our cDNA targets. Our platform would be a yeast ORF array, containing DNA oligonucleotide probes which are designed to hybridize to sequences specific to each gene in the yeast genome.

Our goal in designing this microarray experiment and analyzing our results will be to identify genes which are differentially expressed in the pre- and posttreatment samples.

2. Experimental Design: Single-Sample Versus Competitive Hybridization

From a data analysis perspective, one of the most important decisions you will need to make when planning your microarray experiment is how to setup your sample hybridizations. In single-sample, or ''one-channel,'' experiments each biological sample being assayed is individually hybridized on an array. In a competitive hybridization, or ''two-channel'' design, two or more biological samples with unique dye-labels are mixed and hybridized together on an array. Typically, your choice of array platform will be tied to your preferred hybridization strategy.

2.1. Single-sample hybridization

A single-sample design can be appealing because it is both conceptually simple and analogous to other common DNA or RNA detection procedures (e.g., northern blot analysis). In these experiments, target molecules from each biological sample are labeled with fluorescent dye and hybridized to the surface of a single array. The intensity of the label associated with each probe is quantified using digital image analysis and these intensity values are used in all subsequent manipulations to represent the quantity of each target in the starting sample. In order for label intensities to be comparable between probes, the labeling procedure must operate at a similar efficiency on each target probed by the array. This means that labeling must be independent of the sequence length or base composition of the target. To meet this requirement, end-labeling schemes are commonly used, whereby a single fluorescent molecule is covalently linked to targets. The major advantage of single-sample designs is that there is a direct, and under optimal conditions potentially linear, relationship between the concentration of each target in the original biological sample and the label intensity measured on the associated probe.

If we were to analyze the biological samples in our simple case study above using a single-channel scheme we would hybridize end-labeled cDNA copies of each of our original biological RNA samples onto individual yeast ORF arrays. The absolute dye intensity imaged on each spot would be used to represent the quantitative expression level of the associated gene target. We could then compare expression levels between samples mathematically. For example, we might ask what the fold-change in expression levels of each transcript is between a pre- and posttreatment sample by comparing probe intensities observed on two different arrays.

2.2. Competitive hybridization

While single-sample designs attempt to measure the *absolute* levels of probed DNA or RNA species, competitive hybridization designs measure the *relative* concentration of each probed species in two or more samples. In standard competitive hybridization designs, two biological samples, usually a ''reference'' or control sample and an ''experiment'' or treatment sample, are labeled with two different dyes. The samples are then mixed and hybridized on the surface of an array. Two digital images are acquired for each array using a wavelength filter specific to the emission spectrum of each of the two dyes. During image analysis, the ratio of the two dye intensities is calculated for each spot. This ratio value is then used in all subsequent analysis steps to represent the relative level of targets in the two starting samples.

Competitive hybridization designs can be more difficult for those new to microarrays to understand, because each measurement is inherently relative,

rather than absolute. However, competitive hybridization also offers a number of significant advantages over single-sample designs. Competitive hybridization schemes tend to be insensitive to differences in the relative efficiency with which different target sequences pass through the sample preparation protocol and are detected on the array. In a typical gene expression experiment, for example, we can imagine that the efficiency of RNA isolation, cDNA synthesis, dye-labeling, and probe hybridization could each be affected by the length and base composition of each target. Because competitive hybridization arrays measure the relative levels of the same target sequence from two different starting biological samples on each probe, differences in the performance characteristics of individual target sequences are not as relevant to the measurements being made.[2]

One disadvantage to competitive hybridization designs is that, with two samples used per array, there is twice the material cost for each. A second consideration to be aware of is that no two dyes will perform identically in an array experiment. In two-color experiments using Cy3 and Cy5, for example, there is often a ''green'' or Cy3-shift among low-intensity spots. For this reason, it is important to flip the association of dyes and samples in replicate array experiments (''dye-flipped'' replicates), to ensure that observations are not the result of a dye-intensity bias. In Section 4, we will also discuss computational methods which allow us to assess and mitigate dye-intensity bias.

Competitive hybridization designs are not limited to two samples. Conceptually the only limit to the number of samples that can be put on a single array is the number of unique dye wavelengths that can be reliably differentiated. Several commercial scanners now support up to four dyes and several data analysis packages are already prepared to handle arbitrarily large numbers of dye-channels.

2.3. Choosing the best approach

Both of these hybridization strategies have been used in the microarray field to produce accurate and reproducible data. A good litmus test for which approach is best suited to your study may be to consider whether or not your question of interest is *comparative*. In the case study introduced above, our biological question is indeed comparative: we are interested in measuring changes in the relative abundance of transcripts in an experimental and control sample. We have a clear reference, the pretreatment sample, to which we want to compare transcript abundance in posttreatment samples.

[2] This is not to suggest that probe intensity is not a consideration in competitive hybridization experiments. As we discuss below, for example, it is important to be aware of dye-specific intensity biases. However, with proper normalization, competitive hybridization designs can be extremely robust across a very wide range of probe intensities.

In this example, the competitive hybridization approach allows us to perform this comparison directly, rather than as mathematical manipulation, thereby simplifying our data analysis approach and avoiding the potential propagation of error.

The advantage of competitive hybridization designs for comparative studies becomes even stronger when we consider investigations involving more than one experimental factor. Suppose we are interested in testing both a wild-type (WT) and mutant strain in our example stress experiment. In a first pass experiment, we might hybridize a pre-stress sample against a post-stress sample for each of these two strains, on two different arrays. How would we interpret our results if we observed subtle differences between the two strains, which could plausibly be either biologically meaningful or simply due to random variation? In a competitive hybridization design we can add a third array into the mix which makes this comparison directly, hybridizing poststress samples from the WT and mutant strains on the same array. Because at least one of two original biological samples has been hybridized onto each of these three arrays we have a powerful tool for ensuring that the observations made on each array are directly comparable.[3] Linear analysis packages such as Limma, discussed in Section 6, allow us to use competitive hybridization designs such as this to rigorously assess the statistical significance of apparent changes in expression level of a given target between two samples.

3. Image Analysis

The computational phase of a microarray experiment begins with the analysis of digital images. Image analysis usually encompasses the following steps:

(1) Identify regions in the image which represent spots where probes have been printed.
(2) Calculate the average intensity of pixels within each spot (foreground pixels).
(3) Calculate the average intensity of pixels which lie outside of spots (background pixels).
(4) Associate spots with platform annotations (probe identifiers).

In this section, we will consider the information held in image files and how it is used to describe the quantity of target material hybridized to probes on an array.

[3] In performing this experiment, we would actually use at least six arrays to obtain data from dye-flipped replicates of each experimental contrast. We would probably also perform a fourth type of array, comparing prestressed WT and mutant samples to isolate the effect of the mutation alone.

3.1. What is a digital image?

To fully understand the processing of microarray images, it is important to understand how information is stored in a digital image. The simplest digital image formats store a table of intensity values corresponding to each pixel position in the image. The resolution of an image is equal to the dimensions of this table, usually expressed as the ''number of columns × the number of rows,'' for example, ''1024 × 1024.'' Microarray images are gray-scale, meaning that each pixel has a single intensity value associated with it. The range of intensity values for each pixel depends on the bit-depth of the image. Typically, microarray images are 16-bit, meaning that pixel intensities can range in value from 0 (black) to 65,535 (white).[4] Do not be alarmed if you open a microarray image in your favorite image viewer and it appears to be blank. Microarray images often fail to render properly in photo applications because these software packages are usually designed to handle only 8-bit images.

When working with microarrays, it is important to be aware of the file format being used to store array images. Microarray images should always be kept in loss-less formats. Compressed image formats, such as JPEG, discard pixel information during compression in the interest of decreasing overall file size. Although it might be tempting to choose a compressed format for long-term storage, because microarray image files can be quite large, storing images in a lossy format is the digital equivalent to throwing away data.

The tagged image file format (TIFF) is a commonly used to store microarray images. By default these files store data uncompressed. Each TIFF can hold an arbitrary number of images, called layers. This feature is used in competitive hybridization experiments to save the images from both dye-channels in a single file. As the name suggests, TIFFs can also hold a set of user-defined tags. Microarray scanner software will often save information about the image acquisition session to these files such as the scanner temperature and laser settings. Other common file formats which offer loss-less compression include PNG (portable network graphics) and GIF (graphics interchange format).

3.2. Data files

Input. Image analysis consumes two types of data: the digital images themselves and platform-specific probe annotations. Probe annotations are usually saved in text files which associate spot addresses on the array with information about the probes printed on each spot. Examples include

[4] Each bit in the file has one of two values (binary), so with 16-bits per pixel this is 2^{16} possible values.

GenePix GAL files and Affymetrix CDF files. These files allow us to link each observation in the array data to:

(1) a spot location, or set of locations, on the surface of the array (information that can be used in quality assessment)
(2) a probe sequence (necessary for MIAME[5] compliance).

Output. Image analysis produces a table of information describing various properties of each spot on the array. Usually these files have as many rows as there are spots on the array. Example formats include GenePix GPR files, Affymetrix CEL files, and SPOT files. Average intensities of the pixels associated with each spot are extracted from these files in the preprocessing step. Information stored in these files often describes the shape of each spot and variation in pixel intensities, both of which can be used to flag low-quality spots during spot filtering.

3.3. Software tools

3.3.1. Commercial packages

Most commercial microarray scanners are sold with accompanying licenses for image analysis software. For example, GenePix is licensed with Axon scanners and offers an integrated image analysis solution, including control of the scanner to acquire images, import of probe information, spot finding, and pixel intensity analysis. One advantage of commercial image analysis packages is that they tend to offer integrated, user-friendly interfaces. Unfortunately, it can be quite costly to obtain a license for these packages in the absence of a hardware purchase. Also, as with any closed-source software, it is usually not possible to obtain detailed information about the implementation of image analysis algorithms.

3.3.2. Open-source and freely available packages

Although none of the free solutions have reached the maturity of most commercial packages, there are several projects worthy of note:

ScanAlyze (http://rana.lbl.gov/EisenSoftware.htm)	An open-source microarray image analysis package written and maintained by Michael Eisen. (Windows only)
MicroArray_Profile (http://www.optinav.com/imagej.html)	An ImageJ (http://rsbweb.nih.gov/ij/) plug in that can be used to analyze microarray images. MicroArray_Profile allows you to define a spot

[5] See Section 8.2.

	grid, automatically adjust spot diameters and export labeled mean pixel intensity values. (Cross-platform)
Spot (http://www.cmis.csiro.au/iap/Spot/spotmanual.htm)	An R package that can be used to perform microarray image analysis, originally written by Yang *et al.* (2001). Installation instructions and a step-by-step user guide are both available on the web site. (Cross-platform)

4. Preprocessing

Data preprocessing can encompasses a number of different manipulations of the raw probe intensity values obtained from image analysis. In general, the goals of preprocessing are to ensure that:

(1) Observations are comparable between different probes on a single array.
(2) Observations are comparable between arrays.
(3) Low-quality observations are removed from the dataset.

Preprocessing procedures are dependent on the array platform and hybridization scheme. In the sections that follow we will focus our discussion on preprocessing data from competitive hybridization experiments. In general, a preprocessing workflow will consume data from image analysis files (GPR, CEL, SPOT) and ultimately produce a table which associates a single expression value with each target across each unique experimental condition. This table then serves as the starting point for higher order analysis such as hierarchical clustering or differential expression analysis.

4.1. Software tools

The preprocessing procedure is often the most computationally intensive and data rich stage of a microarray analysis scheme, usually encompassing several transformations of the primary data. Because of this, it is important to establish a set of best practices for your lab which ensure that the preprocessing of data are both consistent between arrays and well documented.

Here we review several useful software tools which can be used to perform preprocessing steps. Section 8 at the end of this chapter addresses

data persistence tools which can be used store and replicate the input to, and output from, your preprocessing procedure.

4.4.1. Spreadsheets

Common spreadsheet applications include Microsoft Excel, OpenOffice Calc, and Google Docs Spreadsheets. Spreadsheet applications are appealing working environments because they are familiar to most researchers and microarray data structures fit neatly into two-dimensional tables. Spreadsheet applications usually feature some basic declarative programing facilities, for example, allowing the user to calculate values using predefined formula based on the data held within a range of cells in a table. There are several common pitfalls of spreadsheet software; however, which should be considered before committing to using spreadsheets for the bulk of a data analysis scheme. First, many spreadsheet applications (with the exception of Google Docs) allow the user to perform sorting and filtering operations on arbitrary subsets of rows or columns based on the current user selection. In a single "click" this can lead to disastrous consequences, such as jumbling ratio values and probe labels. Second, spreadsheet packages usually do not build a long-term record of changes made to the data by the user (again, with the exception of Google Docs), placing the onus of manually recording each and every data manipulation on the researcher. Finally, many spreadsheet applications impose hard limitations on the number of columns and rows present in each table. Before committing to a particular software package it is important to verify that the software will support the dimensions of your array platform and the number of arrays you intend to perform.

4.4.2. Commercial statistics packages

Commercial statistical packages such as SPSS or Minitab offer a step-up from basic spreadsheet applications. These software packages are usually designed to efficiently handle large datasets and offer more robust facilities for describing the structure of a tabular dataset than most spreadsheet software. These packages also implement a broader range of statistical algorithms and usually support more sophisticated programing capabilities. Disadvantages of these software packages include the expense of licenses and, because they are closed-source, a lack of access to details about the implementation of the statistical methods they feature.

4.4.3. [R] and Bioconductor

In the domain of microarray data analysis one open-source statistical environment is worthy of special note: R.[6] R is a freely available, open-source derivative of the S-Plus system and has attracted the attention of wide range

[6] http://www.r-project.org/

of users in both the academic and private sectors. There are several features which make R appealing as a data analysis environment. First, it is entirely open-source. This means that the software is both free to obtain and that the implementation details of all statistical methods found in R are freely available and open to scrutiny. Second, R has emerged as one of the most accommodating environments for statistical research, leading to the establishment of an active and innovative community. Finally, R implements a full-featured programming language which can be used to abstract and automate sophisticated analysis procedures.

Because of these features, R was chosen as the host environment for the Bioconductor[7] suite of microarray data analysis software (Gentleman *et al.*, 2004). Bioconductor boasts an impressive array of microarray analysis tools which support a wide variety of platforms and address both preprocessing and higher order analysis. Although R and Bioconductor offer an ideal set of microarray data analysis tools, the initial learning curve can be quite steep, especially for those unfamiliar with command-line environments or scripting languages. However, if your microarray project extends beyond a small number of arrays, spending the time to learn how to use R and Bioconductor at the start of a project can save you many hours of work down the road. In contrast to working in a spreadsheet application, once you have designed a data analysis procedure in R you will never have to manually perform it again: any set of commands can be saved in a script file and run on new input data.

To get started with R, one-click installers are available for Windows, Mac OSX, and Linux platforms and can be found in the "download" section of the R-project.org web site. When you launch R on your system you will be presented with an "interactive interpreter" window in which you can enter commands. R comes packaged with a number of documents in PDF format aimed at the new user. The best place to start is with the introductory "R-intro.pdf." On Windows, users can find this document from within the "Rgui" application (found in the Start menu after installation) by opening the "Help" menu, selecting "Manuals (in PDF)," and then "An Introduction to R." The exercises in this document should take new users a few hours to work through and will introduce you to all of the key features and concepts needed to implement data analysis schemes in R.

Installing Bioconductor from within the R environment is easy. To perform a standard installation, enter the following commands at the R prompt (denoted below as a ">"):

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

[7] http://www.bioconductor.org/

Additional installation options and instructions for installing nonstandard packages are available on the Bioconductor web site.[8] Once installed, you can load a specific Bioconductor package with the library( ) function. To load the ''marray'' suite of preprocessing functions, for example, enter:

> library(marray)

The R help() function can be used to obtain information about the use of methods implemented in Bioconductor packages. For example, the following command will display information about the usage of the main array normalization function (''maNorm'') in the ''marray'' package:

> help(maNorm)

Many Bioconductor packages also come packaged with tutorial style documents in PDF format called ''Vignettes.'' To access the vignettes from within R:

> library(Biobase)
> openVignette( )

You can then enter the number corresponding to the vignette of interest and the associated PDF should open on your system. For microarray normalization procedures the ''marray'' and ''Limma'' vignettes are great places to get started.

4.2. Calculating ratio values

In a competitive hybridization experiment, ratios are calculated for each spot from the average pixel intensities in each channel. There are several methods which can be used to calculate the average pixel intensity for a spot. The first option to consider is which descriptive statistic will be used, usually the mean or median. On a high-quality spot, the values of the mean and median pixel intensities should be very similar. One advantage to choosing the median pixel intensity over mean is that it will be more robust to small numbers of outliers, which can help to mitigate the effects of small scratches (aberrantly low-intensity pixels) or dust (aberrantly high-intensity pixels). The second consideration is whether the channel ratios are calculated before or after averaging. For example, one can first average the intensity of pixels in the red and green channels independently and then calculate the ratio of these two averages (''ratio of the means'' or ''ratio of the medians''). Alternatively, one can calculate the ratio of the intensity of the red and green channel for each pixel and then average the set of ratios

[8] http://www.bioconductor.org/docs/install/

(''mean of ratios'' or ''median of ratios''). Again, for a high quality spot, we would expect these four values to be similar.

By convention, we usually assign the ''red'' (Cy5; 635 nm filter) channel to the experimental treatment and ''green'' (Cy3; 536 nm filter) channel to the reference sample, so that this ratio is greater than 1 when a gene target increases in abundance in response to a treatment, and the ratio is less than 1 when a target decreases in abundance. It is important to remember to ''flip'' values, or calculate the inverse of the ratios obtained from image analysis, for arrays in which the experimental and reference channels are arranged in the reverse orientation.

Finally, it is also common practice to transform ratio values to a linear scale. A $\log_2$ transformation makes ratios particularly convenient to work with because it is simple to conceptualize the fold-change in a ratio given a $\log_2$ value. For example: $\log_2(2/1) > 1$, $\log_2(1/1) > 0$, and $\log_2(1/2) > -1$.

4.3. Normalizing ratio values

In a competitive hybridization experiment, the total signal intensities of the red and green channels will never be perfectly balanced. As described in Chapter 3 in this volume, consideration is given to signal balance in both the sample preparation and image acquisition stages. However, a final mathematical manipulation of the data is required to account for any remaining array-specific biases in signal intensity in order for ratio values to be fairly compared between different arrays. Many normalization strategies also include adjustments for technical bias to improve comparison between probes on the same array.

It is important to carefully consider both the structure of your data and the underlying biological question of interest when choosing a normalization scheme. For best practices, it is recommended that plots be made which will allow you to assess the overall distribution of ratio values on each array before and after normalization. Histograms or density plots such as those illustrated in Fig. 2.2 are easy to make in spreadsheet software and statistical packages. Boxplots are useful when you want to compare the distribution of ratio values across a large number of arrays on a single axis (Fig. 2.3). Finally, scatter plots comparing ratio values to spot pixel intensity (in Bioconductor, ''MA'' plots) are useful for assessing the degree to which spot ratios have been influenced by a dye-intensity bias.

4.3.1. Global mean or median normalization

The simplest form of ratio normalization is a global mean or median adjustment. As illustrated in Fig. 2.2, in this normalization procedure each ratio is adjusted by a constant value to center the mean or median of the distribution of ratios observed on the array. In this example, the ratios on

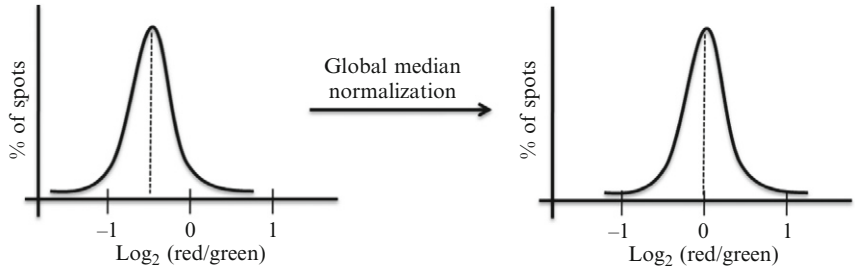

Figure 2.2 A hypothetical distribution of $\log_2$(ratio) values before and after global median normalization. Before normalization this is a ''green'' array, where the average ratio of red/green is <1. After normalization we have shifted the distribution to the right, moving the median $\log_2$(ratio) value to 0 and preserving the shape of the distribution.

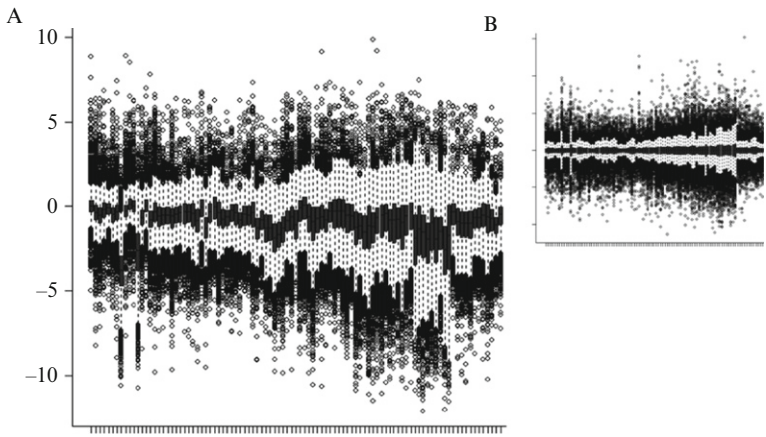

Figure 2.3 Boxplots showing example distributions of $\log_2$(ratio) across a large number of arrays. Log_2(ratio) values are shown on the *y*-axis and data from each array is plotted in a single boxplot along the *x*-axis. Distributions are shown before (A) and after (B) normalization. Notice that we can easily identify two low-quality arrays as outliers on the left-hand side of these graphs.

our array showed a clear ''green'' bias ($\log_2(\text{ratio}) < 0$), which we can account for by adding a constant value to the $\log_2$ transformed ratios (or by multiplying by a constant value if we are working with raw ratios).

A global adjustment is appropriate if we can assume that the total amount of input material from the experimental and reference samples *should be* equivalent. In most cases a median adjustment is preferable to a mean adjustment, because it will be insensitive to outliers. This adjustment is easy to calculate in a spreadsheet or statistical package.

4.3.2. Normalizing to spike-in controls or housekeeping genes

In cases where we expect a large proportion of targets to show a biologically relevant change in expression levels between the experimental and reference samples, a global median normalization may not be appropriate. The alternative is to normalize features on the array based on the behavior of a small subset of spots we expect to not show expression level changes between the two samples. One option is to choose probes which target "housekeeping" genes which we can reasonably assume will not be affected by the treatment. A second strategy is to design probes which do not target any of the DNA/RNA species in the original biological sample, but rather hybridize to a "spike-in" control which can be added to the samples at a standard concentration. One disadvantage to both of these schemes is they rely heavily on the behavior of a relatively small number of spots, and the consistency of the underlying concentration of material probed by those spots.

4.3.3. Adjusting for intensity bias

A global normalization, based either on the median ratio value or control spots, adjusts all ratios using a constant value, irrespective of the signal intensity observed on each spot. However, most competitive hybridization arrays exhibit a bias in ratio value across different signal intensities. As mentioned previously, one common cause for this bias can be variability in the behavior of different dyes at different levels of intensity. LOESS (local polynomial regression fitting) is a regression-based approach which can be used to adjust ratio values based on the observed relationship between spot ratio and intensity (Cleveland *et al.*, 1992; Yang *et al.*, 2002). An excellent application of this procedure to microarray data is provided by Bioconductor "marray" package.[9]

4.3.4. Adjusting for spatial or print-tip bias

Another source of technical error in microarray data which normalization can be used to minimize is bias in ratios arising from the physical location of spots on the array. Depending on the array platform, location bias can be caused by the manufacturing process (e.g., printing efficiencies of different print-tips in robotically pinned arrays), the hybridization process or inconsistencies in scanner alignment. Although it is possible to manually implement spatially aware normalization procedures using common spreadsheet software, Bioconductor offers a number of well developed and convenient tools. The marray package, for example, can be used to apply LOESS smoothing to ratios based on either printing block (for pinned arrays) or the two-dimensional spatial bias in the local neighborhood of each spot.

[9] See documentation for the "maNorm" wrapper function.

The Bioconductor "arrayQuality"[10] package also offers powerful visualization tools for assessing spatial bias across the surface of an array.

4.3.5. More aggressive approaches

If your biological question of interest and the structure of your data conform to a stricter set of assumptions than standard expression arrays, there are a variety of more aggressive normalization techniques which can be used. For example, if you can assume that the variation in ratio distributions should be equivalent between arrays you may consider performing scale or quantile normalizations (Dudoit and Yang, 2002).

4.4. Quality assessment, filtering, and handling replicates

A data analysis procedure is only as strong as the quality of the data fed into it. It is, therefore, important to assess the quality of your array data and implement an effective filtering scheme. Quality assessments can be both qualitative and quantitative, and can be used to filter individual data points or entire arrays.

4.4.1. Spot quality

The first stage of spot filtering occurs during image analysis. Spots which cannot be found by the image analysis software should be flagged for downstream filtering. Spots should also be flagged if they are smeared, continuous with another spot, significantly marred by dust, or scratched. Depending on your array platform, it may also be useful to automatically flag spots which are bigger or smaller than certain reasonable size cutoffs or for which there are extremely high variances in the pixel intensities in either dye-channel. Flags can be added to image analysis data tables using a calculated column in a spreadsheet or with logical index vectors in R. Generally, flagged spots are excluded from ratio normalization calculations.

4.4.2. Array quality

As a general rule of thumb, if an array "looks bad" then it probably is. The Bioconductor arrayQuality[10] package implements a number of plotting techniques which can be used to identify spatial bias across the surface of an array. For example, a heatmap of ratio-value ranks on an array will readily reveal any spatial bias in spot ratios. If your platform design includes replicate spots (identical probes printed in multiple locations) which are distributed across different locations on the array it is possible to assess whether or not variation in ratio values can be explained by surface position. Replicate spots

[10] http://bioconductor.org/packages/2.4/bioc/html/arrayQuality.html

also allow you to calculate an average variance among replicate groups, a parameter which can provide a useful measure of array quality.

4.4.3. Utilizing spot and array replicates

There are a number of ways in which technical replication can be used to enhance the power of microarray-based investigations. The central question of how to best make use of technical replication comes down to when in the analysis procedure replicates should be averaged and how to utilize information about the variability underlying each unique measurement. Spot replicate averaging can be accomplished by taking the mean or median. Whenever you average values, it is also important to calculate a measurement of variation among the observations contributing to each average. Variance and standard deviation are used most commonly. As noted above, spot replicate and array replicate variation can be used as a quantitative measure of array quality. Information about the variation among replicates can also be used to ''weight'' observations in most clustering algorithms (see Section 5), allowing higher quality observations to have a stronger influence on the structure of the resulting graph. Finally, replicate variation is also used in differential expression analysis (see Section 6). When submitting datasets to public repositories (see Section 8.2), it is important to include preaveraged data, so that interested parties can independently recreate these data analysis steps.

4.5. Preprocessing Affymetrix arrays

Although the goals for preprocessing single-sample arrays are similar to those for competitive hybridization arrays, the preprocessing approaches differ significantly. Single-sample arrays, such as the Affymetrix GeneChip platform, often include a series of ''Perfect match'' (PM) and ''Mismatch'' (MM) probes for each gene target. Preprocessing of these arrays involves summarizing PM and MM probe set behaviors. There are a number of software packages available which implement different preprocessing algorithms for Affymetrix arrays. Two mature and popular options are:

Expression console	Affymetrix offers their expression array analysis software free of charge for registered users on their web site (http://www.affymetrix.com/support/technical/software_downloads.affx). The latest generation of this analysis suite implements several preprocessing algorithms including MAS, PLIER, and RMA.
Bioconductor	The Bioconductor project includes a number of packages which can be used to normalize data from Affymetrix arrays, including: ''affy,'' ''gcrma,'' and ''affyPLM'' (Bolstad *et al.*, 2003; Irizarry *et al.*, 2003).

5. Visualizing Data Using Cluster Analysis

One of the most popular ways to explore microarray datasets is with clustering analysis. Microarray datasets may contain information about the behaviors of ~10k–1M different probes across dozens or hundreds of different experiments, resulting in a grid of data which is far too large to simply "browse." Clustering techniques are used to order the data according to the behavior of probed targets (genes), experimental factors (arrays), or both. Visualization tools are then used to explore the resulting data structure.

5.1. Hierarchical clustering

The most common clustering algorithm used in the microarray field is hierarchical clustering. Hierarchical clustering is an uncensored machine learning method, meaning that it is used to order a dataset based on the data alone rather than a predefined model.

5.1.1. An example use-case

Let's consider our microarray case study from the introduction. After we have analyzed our array images, normalized and $\log_2$ transformed our ratio values, and filtered out low-quality data we will be left with a table of information which describes the behavior of target genes in each of our biological sample hybridizations (Table 2.1). We can use hierarchical clustering of this table of values to help us identify genes which exhibit similar behaviors across this stress time course. In this case, we would only want to cluster the gene or target axis, because the experiments already have an obvious biologically meaningful order.

5.1.2. How hierarchical clustering works

Hierarchical clustering begins by examining a list of elements, in this example a list of genes, to identify the two elements which are most similar. The results of a hierarchical clustering run are highly dependent on the way in which "similar" is defined. Pearson correlation is a straightforward similarity metric to imagine in this context: the similarity between any

Table 2.1 Example data structure produced by preprocessing which can be used for cluster analysis

	Array 1 (stress time point 1/control)	Array 2 (stress time point 2/control)	...
Gene1	−0.1503	−0.3861	...
Gene2	0.3857	0.2168	...
...	...	...	...

two genes can be calculated as the Pearson correlation of $\log_2$(ratio) values observed across the arrays in the time course. The absolute value of the Pearson correlation could be used if we wanted genes which showed large changes at the same time point to score as similar, irrespective of whether expression levels went up or down. Nonparametric measures such as rank order or Euclidean distances can also be used to evaluate similarity. Choosing a different similarity metric can dramatically affect the structure of a cluster, so it can be beneficial to spend some time exploring your options.

Once the two most similar genes have been identified in the starting list, they are associated on a single branch of a tree. The original data associated with these two genes is then removed from the list of elements and replaced with a new entry that represents the "average" behavior of both. The method by which the "average" behavior of a branch is represented in subsequent similarity calculations (the linkage method) is the second major parameter which controls the behavior of hierarchical clustering.

5.1.3. Common pitfalls

There are several features of hierarchical clustering algorithms which can lead new users astray in the interpretation of their results. First, although hierarchical clusters are often used to "group" genes or experiments into discrete subsets, grouping is not the goal of hierarchical clustering *per se*. Commonly used hierarchical clustering algorithms only operate on the pair-wise distances between elements in a list; they do not consider the larger structure of the data. As such, these algorithms are "bottom-up" approaches and do not necessarily create a tree in which the average distance between all elements is fully minimized, nor do they suggest a best cutoff level in the resulting tree to produce meaningful subsets of elements. There are, however, a number of techniques which have been developed to help overcome these limitations of hierarchical clustering, notably the tree "pruning" algorithms implemented in the Bioconductor "hopach" library (van der Laan and Pollard, 2003).

When exploring a hierarchical cluster, it is important to remember that each node has two equivalent orientations and that the orientation chosen when a tree is initially rendered is effectively arbitrary. Flipping the orientation of a node can dramatically change the visual appearance of a cluster, because of changes in the linear order of the gene or array axis, but has no effect on the overall pair-wise distances between genes. When considering the relatedness of elements on a clustered graph it is important to pay close attention to the actual distance between elements in the tree rather than the relative order on the plot.

5.1.4. Software

The open-source Cluster 3.0 package features an easy-to-use interface and a host of clustering procedures (de Hoon *et al.*, 2004). Cluster 3.0 can read data from tab-delimited text files produced in the preprocessing step.

Depending on the setup, clustering runs produce files with some or all of the following extensions:

.cdt	These files hold the original table of data, with targets (genes) listed in each row and sample comparisons (experiments/arrays) listed in each column. The CDT tables may also have a gene weight (GWEIGHT) column and experiment weight (EWEIGHT) row which describe the clustering weight of each gene and experiment, respectively. As mentioned above, it can be useful to draw weights for a clustering run from measures of variance among technical replicates.
.gtr	Gene-tree files contain tables which describe each branch in the tree as an association of two child genes/nodes (1st and 2nd column) and a parental gene/node (3rd column), as well as the distance from children to parent (4th column).
.atr	Array-tree files are identical to gene-tree files, but describe array clustering.

The information saved in these output files can be visualized using the open-source, cross-platform, Java TreeView package (Saldanha, 2004). Java TreeView draws a heatmap of ratio values from data saved in the CDT file and associated gene- or array-trees if similarly named GTR or ATR files are found in the same directory.

5.2. Partitioning and network-based approaches

In addition to hierarchical clustering, there are a wide variety of other types of clustering methods which can be used to explore microarray datasets. Several commonly used partitioning methods include: self-organizing maps (SOMs) (available in Cluster 3.0 and the ''som'' R package), *k*-means clustering (available in Cluster 3.0), and Prediction Analysis for Microarrays[11] (PAM) (available as the ''PAM'' R package and as an Excel plug in). Like hierarchical clustering, these partitioning algorithms make use of distance metrics and linkage methods to cluster closely associated genes or arrays. Unlike hierarchical clustering, partitioning algorithms are designed to construct defined subsets of element, although for some approaches, like SOMs, the use of stochastic parameters means multiple runs on the same dataset may not always produce the same result.

Among network-based approaches, the BioLayout Express[12] implementation of the Markov Cluster Algorithm (MCL) is particularly worthy of note (Freeman *et al.*, 2007). MCL belongs to a class of algorithms which

[11] http://www-stat.stanford.edu/~tibs/PAM/
[12] http://www.biolayout.org/

optimize the overall distances between nodes across the entire set of elements being clustered. Network-based approaches are much more flexible than hierarchical clustering in the kinds of paths between nodes which can be drawn, and can be more conducive to a visual exploration of gene groups. BioLayout Express is an open-source, cross-platform software package with excellent documentation and a well-developed user-interface.

6. Assessing the Statistical Evidence for Differential Expression

Microarray data are often used to identify changes in gene expression in response to an experimental treatment. Conceptually, we would like to be able to extract a unique list of genes from a microarray dataset which are differentially expressed in one biological sample compared to another. The challenge comes when deciding how to draw a cutoff in ratio values. Although a blanket cutoff of a ''twofold change'' has been used in the microarray field in the past, this is not a statistically rigorous approach nor is it a reasonable approximation for many datasets.

Our goal in assessing the evidence for differential expression of targets in a microarray dataset can be conceptualized in the same terms as any canonical hypothesis testing problem. In this case, the null hypothesis, which we want to know whether or not to reject, is that the observed $\log_2$(ratio) value for a particular target is actually no different than 0. Given the variability in the measurements observed among replicates, and the average expression change across the dataset, we want to calculate statistics which will allow us to assess the probability that accepting or rejecting this null hypothesis will result in a false-positive (type I error; identifying genes as differentially expressed which are not) or false-negative (type II error; identifying genes as not differentially expressed which are) determination. However, calculating meaningful significance levels for each gene in a large dataset is a complex problem. For example, if we test this null hypothesis for 15,000 probes on an array using a classical *t*-test, the high level of multiple testing shifts the scale of meaningful *p*-values away from what we are normally used to considering.

Here we briefly discuss two statistical approaches to this problem which are implemented in mature software packages and have been used to great effect in yeast studies:

6.1. Significance analysis of microarrays

Significance analysis of microarrays (SAM) uses gene-specific *t*-tests, calculated using a nonparametric statistic, to provide an estimate of the false discovery rate at a given ratio value cutoff (Tusher *et al.*, 2001). SAM is

flexible enough to handle most common experimental designs, but is less versatile in this regard than Limma. SAM is available as both an [R] library and a Microsoft Excel plug in. It is free to download for academic users.

6.2. Limma

Limma is a Bioconductor software package which uses linear models to analyze microarray data and assess evidence for differential gene expression (Smyth, 2004). Limma can be used to model virtually any experimental design, accounting for sample measurements taken across complex sets of competitive hybridization pairs. An extensive user manual and step-by-step walkthrough are provided in the Limma documentation.[13]

7. Exploring Gene Sets

Having identified an interesting group of targets (e.g., genes) using cluster analysis or differential expression assessment, the next question we usually want to address is: what is similar about the members of each group? Here we review some common approaches to this question.

7.1. Gene Ontology term mapping

The Gene Ontology (GO) project is cross-species gene annotation effort which defines a controlled vocabulary of terms describing a gene product's function, biological process, or cellular component (Ashburner *et al.*, 2000). When presented with a subset of genes which show a similar pattern of expression in a microarray dataset, it can be useful to determine whether or not the GO annotations associated with those genes suggest a common biological function. In GO, terms are related to one another through a branched hierarchy.[14] Genes can be annotated with any number of terms from any level of this hierarchy. Conceptually the complex structure of GO terms is appealing, because it allows for a high degree of flexibility in gene labeling. Unfortunately, this structure also makes it difficult to appropriately estimate the statistical relevance of a potentially overrepresented GO term in a list of genes. The GO Slim Mapper[15] hosted at the Saccharomyces Genome Database offers a Web-based solution which assesses the statistical likelihood that a GO term is meaningfully overrepresented in a given set of genes, plotting the results on a GO term graph. The Bioconductor

[13] See "userguide.pdf" in the "/doc" subdirectory of the limma library or enter "> library(limma); limma UsersGuide( )" at the R prompt.
[14] More properly, the GO topology is an acyclic graph as child terms can be associated with multiple parents.
[15] http://www.yeastgenome.org/cgi-bin/GO/goSlimMapper.pl

"GOSemSim" package[16] also provides methods for estimating GO semantic similarities in gene sets. Finally, the GeneMAPP software package offers an excellent set of tools for drawing pathways based on GO terms which can then be highlighted based on gene behavior in a microarray dataset (Dahlquist *et al.*, 2002).

7.2. Motif searching

Motif searching can be used to identify potential *cis*-regulatory elements associated with coregulated gene products. MEME and AlignACE are two popular software packages which each offer Web-based submission (Bailey *et al.*, 2009; Hughes *et al.*, 2000). An in-depth discussion of motif searching algorithms by Hao Li appears in "The Guide to Yeast Genetics and Molecular Biology, Part B" (Li, 2002).

7.3. Network visualization

There are a number of tools available which allow one to visualize interaction networks, integrating microarray data, proteomic data and other sources of gene annotations. The open-source Cytoscape[17] project is particularly worthy of note for its excellent documentation, accessible learning curve, and active community.

7.4. Graphing array data on genome tracks

Analysis of tiling microarray data usually involves visualizing probe behaviors on genomic tracks. The "genome browser" tool in the Gaggle project[18] offers an easy-to-learn software solution for genomic visualization of microarray data. Once probe identifiers are associated with a chromosome number and chromosomal coordinates, microarray data can be visualized with heatmaps, scatter plots, or line graphs on top of genome tracks drawn from the UCSC genome browser. A note of caution for those who usually use SGD as the source of genomic coordinates: the UCSC genome browser draws from the October, 2003 assembly, while SGD has continued to update the reference sequence. This means that there are slight inconsistencies between the genome coordinate systems in these two databases. The SGD Genome Browser (GBrowse) offers a Web-based alternative[19] which uses the current SGD coordinates.

[16] http://bioconductor.org/packages/2.4/bioc/html/GOSemSim.html
[17] http://www.cytoscape.org/
[18] http://gaggle.systemsbiology.net/docs/geese/genomebrowser/
[19] http://www.yeastgenome.org/cgi-bin/gbrowse/scgenome

8. Managing Data

8.1. Data persistence and integrity

One important, and easily overlooked, consideration when setting up a new microarray data analysis project in your lab is the best way to handle storage of microarray data and analysis results. Individual image and data files can be quite large, and data analysis pipelines typically produce a number of files across several stages. Ultimately, published array data will be archived in a public, off-site, MIAME-compliant database (see below), but how should the data associated with moderate- or large-scale microarray projects be handled within the lab?

One approach is to use a relational database to store array data and provide a Web-based front-end for queries and data-submission (e.g., the UCSF NOMAD[20] project). These solutions allow a working group to store array data in a centralized location, facilitating backups and maintenance. Modern, open-source, relational databases such as MySQL[21] are capable of handling extremely large chunks of data, storing array images alongside data tables (such as image analysis files, probe information, clustering results, etc.), and provide efficient data querying facilities.

Often, however, it is more convenient to work with microarray data as files on the local file system, rather than as tables stored in a relational database. Moving data to and from a database for use in other software packages can be cumbersome and confusing for those unfamiliar with these technologies. In my own work I have settled a convenient solution that works well for most types of projects: version control systems.

Version control systems such as CVS and Subversion[22] were originally developed to facilitate software development projects involving many developers. Using Subversion to manage your microarray data files is fairly simple: you work with your microarray data files in a set of folders saved to your local machine and then synchronize the state of these folders with a Subversions server after each major change or manipulation. Version control systems such as Subversion offer several facilities which are extremely useful in a microarray data analysis project. Below we review some of these features, all of which should be taken into consideration when choosing a data storage strategy for you lab.

[20] http://sourceforge.net/projects/ucsf-nomad/
[21] http://www.mysql.com/
[22] http://subversion.tigris.org/

8.1.1. Data replication

In Subversion parlance the files and folders you keep on your local machine are a ''working copy'' of a ''repository'' kept on the server. You can make as many working copies of your files on different computers as you need, each of which can be kept synchronized with the central repository. The concept here is similar to that of a fileserver, but Subversion is more useful in several ways. First, the working copy is actually a local copy of all of your files, so you do not need to be connected to a network to access your data. Second, it is up to you explicitly decide when you want commit local changes to the repository. This allows you to organize sets of changes into logical transactions.

8.1.2. Multiuser support

Many people can work with the microarray data files simultaneously using their own working copies of the archive. When users commit the changes they have made locally back to the repository, Subversion detects conflicts (cases where mutually exclusive changes have been made to a file) and offers a rich set of tools that help you to merge changes into a new version of the file. Subversion servers also allow you to setup user authentication (unique user names and passwords for different member of you lab) and set read/write permissions on different portions of the archive.

8.1.3. Transactions and logs

Subversions servers save *changes* to files in the repository instead of copies of the files themselves. Each time you commit local changes in your working copy to the server, a log entry is created tracking all of the changes that were made. This means that it is possible to revert a working copy, or the central repository itself, to any previous version of the archive. Inevitably, when working on a data analysis project there will come a time when you are unsure of whether or not you have performed an analysis with the intended parameters or when you accidentally overwrite an important set of files. The version control system makes solving these problems straightforward: you can simply revert the repository to the last point in time when you know it was in a sane state. Subversion allows you to save a text log entry alongside every change to the contents of the repository. I have found this to be an incredibly powerful way to keep a record of data analysis projects. Log viewers are available which allow you to browse each of your log entries alongside the associated changes to files in the repository.

8.1.4. Web-based access

The Subversion community has developed a mature Apache web server module which makes it easy to ''publish'' your Subversion repository on the Web. I have found this to be an extremely convenient way to share data analysis files with off-site collaborators.

There are a number of version control systems available, each with slightly different performance characteristics and feature sets. Subversion is a well rounded, open-source, cross-platform solution which I have found works quite well for data analysis projects. Subversion is comprised of two software components: a *client* that needs to be installed on each computer where you want to make a ''working copy'' and a *server* that should be setup on a machine with reliable internet connectivity and a regular backup schedule. If you are using Linux, chances are quite good that both the Subversion client and server are already installed. Installers and binary files for other platforms, including Windows and Mac OSX, can be found on the Subversion web site. For windows users I would highly recommend both TortoiseSVN,[23] which integrates Subversion client facilities into Windows explorer, and VisualSVN[24] which allows you to setup a Subversion server and web server using a simple installer.

8.2. Public data repositories and MIAME compliance

Community standards, and many journal submission agreements, require that microarray data presented in publications be submitted to public databases. Before the development of centralized microarray data hosting centers, many researchers posted microarray files on private lab web sites. This solution is problematic for several reasons. First, Web URLs are not a reliable resource: institutions often reorganize the URL structure of their web sites and labs can change institutional affiliation. Second, the structure and completeness of the available data can be extremely varied, making it difficult for interested third parties to recreate the published analysis or use the data in new studies.

In response to inconsistencies in data-sharing practices in the microarray community, the Microarray Gene Expression Data Society[25] (or MGED) was formed to develop a standard describing the Minimal Information About a Microarray Experiment, or MIAME (Brazma *et al.*, 2001). The MIAME standard takes on the daunting task of articulating a formal set of data structures which describe a wide variety of different microarray experiments and platforms. Because of this, the standard itself is extensive and relies on a relatively abstract nomenclature to remain platform agnostic.

Fortunately there are a number of public, curated, MIAME-compliant databases designed to guide experimenters through the data annotation process. Three popular choices are:

[23] http://tortoisesvn.tigris.org/
[24] http://www.visualsvn.com/server/
[25] http://www.mged.org/

GEO	The Gene Expression Omnibus (GEO) (http://www.ncbi.nlm.nih.gov/projects/geo/) project developed and hosted at the NCBI is a free, full featured, database maintained by a responsive curatorial staff. GEO supports submission of a wide variety of gene expression datasets, offers a user-friendly Web-based submission system, and scales well to handle large submissions. GEO supports timed public release of datasets and the creation of private URLs which can be provided in the peer-review process. Finally, datasets stored in GEO are automatically searched when users enter terms in the "all databases" search box on the NCBI home page.
ArrayExpress	ArrayExpress (http://www.ebi.ac.uk/microarray-as/ae/), developed and hosted at the EBI, is similar to GEO in scope and sophistication. There are some minor differences between the two sites in the searching facilities offered and the batch data upload file formats that are supported.
SMD	The Stanford Microarray Database (http://smd.stanford.edu/index.shtml) offers several Web-based data analysis packages not available from other repositories. However, SMD charges a significant usage fee to labs outside of Stanford University.

For projects with a small number of arrays, submitting data to GEO is as simple as creating an account and following the Web-based guide to upload data files and provide MIAME-compliant annotations. GEO submissions are composed of three types of records:

Platform	The platform record describes your microarray. This includes annotations describing the array substrate, manufacturing process, and probe sequences. If you are working with a common commercial array platform there is a good chance that a record has already been submitted for your array. If this is the case, you do not need to create a duplicate entry; you can simply skip this step and link your samples to the preexisting platform record. It is useful to attach your platform-specific probe/spot annotation table (GAL file, CDF, etc.) as a supplementary file on these records.
Sample	Sample records describe each *hybridization* event in your microarray experiment. For single-sample designs you will create one sample record for each individual array in your

(*continued*)

	experiment. For competitive hybridization designs, you will also create one sample record for each array in your experiment, but in this case each sample record will actually describe *two* biological samples. The sample record combines annotations describing the origin and preparation of the biological sample hybridized on the array, along information about the hybridization conditions and data acquisition procedure. If your data preprocessing scheme involves a normalization step, it is common to provide the results of this normalization in the data table associated with each sample record. Attaching your platform-specific image analysis results (GPR file, CEL, SPOT, etc.) as a supplementary file will allow interested users to explore alternative normalization and preprocessing approaches. Each sample record points to a single platform record.
Series	The series record describes your study and the relationships among the associated sample records. This record is the main entry point for users interested in your dataset. The series record will allow you to describe the unique experimental factors examined in your study and the structure of your experimental replicates. Finally, the series page provides users with a set of links which allow them to download your data in a variety of formats. Each series record points to one or more sample records.

All GEO submissions are reviewed by a curator before they are made public. This ensures that the submitted data and annotations meet a set of minimum standards.

If your microarray project involves a large number of arrays, it may be too cumbersome and time consuming to manually submit data to GEO through the Web-based forms. For these cases, GEO supports a number of batch deposit formats. The simplest of these to use and understand is the SOFT format. SOFT files are plain text files which associate annotations, in header rows, with data, in tab-delimited tables. Each of the GEO record types (Platform, Sample, and Series) can be described in a SOFT file and each field that appears on the Web-based forms is given a corresponding label for use in SOFT file header rows. These files are relatively easy to produce with spreadsheet software or using simple scripts. Extensive documentation is available on the GEO web site.[26]

[26] http://www.ncbi.nlm.nih.gov/projects/geo/info/soft2.html

ACKNOWLEDGMENTS

I thank M. Bergkessel, C. Guthrie, M. Fitzpatrick, and R. Chande for their critical feedback on this manuscript.

REFERENCES

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., *et al.* (2000). Gene ontology: Tool for the unification of biology. The Gene Ontology Consortium. *Nat. Genet.* **25,** 25–29.

Bailey, T. L., Boden, M., Buske, F. A., Frith, M., Grant, C. E., Clementi, L., Ren, J., Li, W. W., and Noble, W. S. (2009). MEME SUITE: Tools for motif discovery and searching. *Nucleic Acids Res.* **37,** W202–W208.

Bolstad, B. M., Irizarry, R. A., Astrand, M., and Speed, T. P. (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics* **19,** 185–193.

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

Cleveland, W. S., Grosse, E., and Shyu, W. M. (1992). Local regression models. *In* "Statistical Models in S," (J. M. Chambers and T. J. Hastie, eds.), Chapman & Hall.

Dahlquist, K. D., Salomonis, N., Vranizan, K., Lawlor, S. C., and Conklin, B. R. (2002). GenMAPP, a new tool for viewing and analyzing microarray data on biological pathways. *Nat. Genet.* **31,** 19–20.

de Hoon, M. J. L., Imoto, S., Nolan, J., and Miyano, S. (2004). Open source clustering software. *Bioinformatics* **20,** 1453–1454.

Dudoit, S., and Yang, Y. H. (2002). Bioconductor R packages for exploratory analysis and normalization of cDNA microarray data. *In* "The Analysis of Gene Expression Data: Methods and Software," (G. Parmigiani, E. S. Garrett, R. A. Irizarry, and S. L. Zeger, eds.), Springer.

Freeman, T. C., Goldovsky, L., Brosch, M., van Dongen, S., Mazière, P., Grocock, R. J., Freilich, S., Thornton, J., and Enright, A. J. (2007). Construction, visualisation, and clustering of transcription networks from microarray expression data. *PLoS Comput. Biol.* **3,** 2032–2042.

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., *et al.* (2004). Bioconductor: Open software development for computational biology and bioinformatics. *Genome Biol.* **5,** R80.

Hughes, J. D., Estep, P. W., Tavazoie, S., and Church, G. M. (2000). Computational identification of cis-regulatory elements associated with groups of functionally related genes in *Saccharomyces cerevisiae*. *J. Mol. Biol.* **296,** 1205–1214.

Irizarry, R. A., Bolstad, B. M., Collin, F., Cope, L. M., Hobbs, B., and Speed, T. P. (2003). Summaries of Affymetrix GeneChip probe level data. *Nucleic Acids Res.* **31,** e15.

Li, H. (2002). Computational approaches to identifying transcription factor binding sites in yeast genome. *In* "Methods in Enzymology: Guide to Yeast Genetics and Molecular Biology," Part B, (C. Guthrie and J. Fink, eds.), pp. 484–975, Academic Press.

Saldanha, A. J. (2004). Java Treeview—Extensible visualization of microarray data. *Bioinformatics* **20,** 3246–3248.

Smyth, G. K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Stat. Appl. Genet. Mol. Biol.* **3,** Article 3.

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

van der Laan, M. J., and Pollard, K. S. (2003). A new algorithm for hybrid hierarchical clustering with visualization and the bootstrap. *J. Stat. Plan. Inference* **117,** 275–303.

Yang, Y. H., Buckley, M. J., and Speed, T. P. (2001). Analysis of cDNA microarray images. *Brief. Bioinform.* **2,** 341–349.

Yang, Y. H., Dudoit, S., Luu, P., Lin, D. M., Peng, V., Ngai, J., and Speed, T. P. (2002). Normalization for cDNA microarray data: A robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Res.* **30,** e15.