MAANOVA: A Software Package for the Analysis of Spotted cDNA Microarray Experiments

Hao Wu[1], M. Kathleen Kerr[2], Xiangqin Cui[1], and Gary A. Churchill[1]

[1]The Jackson Laboratory, Bar Harbor, ME

[2]The University of Washington, Seattle, WA

ABSTRACT We describe a software package called MAANOVA, for MicroArray ANalysis Of VAriance. MAANOVA is a collection of functions for statistical analysis of gene expression data from two-color cDNA microarray experiments. It is available in both the Matlab and R programming environments and can be run on any platform that supports these packages. MAANOVA allows the user to assess data quality, apply data transformations, estimate relative gene expression from designed experiments with ANOVA models, evaluate and interpret ANOVA models, formally test for differential expression of genes and estimate false discovery rates; produce graphical summaries of expression patterns; and perform cluster analysis with bootstrapping. The development of MAANOVA was motivated by the need to analyze microarray data that arise from sophisticated designed experiments. MAANOVA provides specialized functions for microarray analysis in an open-ended format within flexible computing environments. MAANOVA functions can be used alone or in combination with other functions for the rigorous statistical analysis of microarray data.

1 Introduction

This chapter describes the philosophy behind and the function of a software package called MAANOVA, for MicroArray ANalysis Of VAriance. MAANOVA is implemented in both Matlab and R programming environments. We focus our discussion on the Matlab implementation and address minor differences in the two implementations below. The package is a collection of functions that can be employed by a data analyst for the purpose of investigating gene expression data from two-color cDNA microarray experiments. Our goal is to provide a computing environment for microarray data analysis that is open ended and flexible. An expert user of the system should be able to write her own functions and scripted analyses to achieve any desired result. At the same time we provide a core set of functions that can be applied in a routine and pre–scripted fashion by a novice user to carry out a rigorous statistical analysis of microarray data.

MAANOVA functions were developed for the purpose of analyzing both small and large scale microarray experiments with arbitrary design structures ranging from simple two array dye-swap experiments to elaborate loops. Basic concepts of microarray experimental designs are discussed by Kerr and Churchill (2001a). Although we encourage the use of efficient, balanced designs, the MAANOVA software can be applied to analyze data from any microarray experiment that uses more than one microarray to assay a set of samples. Microarray experiments may be implemented for many different purposes and we do not wish to prescribe or in any way limit the possibilities for their application. We believe that scientists should design experiments based on the specific goals of their investigations and statistical principles. Design choices should not be determined by the availability of analysis software.

2 Methods

In this section we introduce the statistical model that is at the core of our approach to microarray analysis. We digress in order to describe examples of experimental designs that will be used to illustrate points below. The remainder of the section follows the sequence of steps in a typical microarray analysis session. The process begins with diagnostics for data quality and transformations of data to remove intensity dependent and spatial effects on relative expression. The next step is fitting the MAANOVA model. We describe the fitting algorithm and some diagnostics that are useful for assessing the quality of the fit. We discuss randomization methods that are available in MAANOVA to assess the significance of statistical tests and to compute interval estimates of relative expression. Although we emphasize randomization techniques, the standard normal theory results are available as well. Lastly, we discuss some clustering techniques that are useful for organizing (long) lists of differentially expressed genes or for investigating relationships among the RNA samples. The application of randomization methods to cluster analysis is a unique and powerful feature of the MAANOVA software package.

2.1 Data Acquisition

The technology underlying cDNA microarray experiments is described in Schena (1999). The raw data from a single microarray consist of a pair of images representing the fluorescent intensities detected by a photo-multiplier tube when the microarray is scanned with each of two lasers. Images are typically stored in a 16 bit TIFF format and data are extracted by segmenting the image and quantifying the intensity associated with each spot. We assume that the experimenter is satisfied with the quality of the images and that the data have been extracted with a software package such as SPOT (Yang et al., 2002) or GenePix (Axon Instruments, 1999). Decisions regarding background subtraction and other adjustments to the raw data are left to the user. Data from a set of microarray slides that constitute an experiment can be assembled into a single flat file for analysis. The input file for MAANOVA requires two columns of intensity values for each array and may include additional columns and header lines as needed. For a four array experiment the 8 columns of intensity data are arranged as $(R_1, G_1, R_2, G_2, R_3, G_3, R_4, G_4)$. We refer to these as the raw intensity data.

2.2 ANOVA models for Microarray data

An analysis of variance model for microarray data was proposed by Kerr *et al.* (2000) and is discussed further by Kerr and Churchill (2001b). Similar models have been described by Pritchard *et al.* (2001) and by Wolfinger *et al.* (2002). The ANOVA model is applied to transformed intensity data, for example, a logarithmic transform of raw intensity data. It allows one to account for sources variation in the data that are attributable to factors other than differential expression of genes, thus it effectively normalizes the data. There is no loss of information, as is the case when raw intensity data are converted to ratios. Furthermore it allows one to combine the information from many different arrays in a single analysis.

In a cDNA microarray experiment, two differentially labeled samples are applied to each array. Thus an experiments with k arrays provides $2k$ measurements for each gene. The key to combining information is to assign each sample a label, which we call the *variety*. All samples from a common source are given a common label and the information about these samples will be combined in the ANOVA analysis.

Denoting the transformed intensity data by $\mathbf{y} = \{y_{ijkgr}\}$ we can express the microarray analysis of variance model as

$$y_{ijkgr} = \mu + A_i + D_j + AD_{ij} + G_g + VG_{kg} + DG_{jg} + AG_{ig} + S_{r(ig)} + \epsilon_{ijkgr}. \qquad (1)$$

The first four terms in the model (μ, A, D and AD) capture the overall average intensity and variations due to arrays, dyes, and labeling reactions (*i.e.*, array by dye interactions) as average effects across all the genes. The term gene G captures the average intensity associated with a particular gene. Although it is tempting to interpret G as representing the average level of expression of a gene, the effect is confounded with specific labeling and hybridization properties of the gene and should be interpreted cautiously. The variety–by–gene terms VG capture variations in the expression levels of a gene across varieties. The VG terms are the quantities of primary interest in our analysis. When the RNA samples (varieties) in an experiment have a complex design of their own (*e.g.*, a factorial design), the VG terms may be structured to reflect the relationships among the samples (Jin *et al.* (2001)). The dye–by–gene terms DG are included to account for a commonly observed technical artefact in which dyes show gene specific effects (Kerr *et al.*, 2002). The array by gene term AG captures the variation of each gene across spots on different arrays. Any gene specific perturbations in the labeling reactions will be captured here. The spot term S captures the differences among the duplicated spots within an array. If there are no duplicated spots, this term is automatically dropped from the model.

The subscripting in equation (1) requires some comment. Arrays are indexed by i, dyes by j and genes by g. The index r is nested within i and g to identify individual spots on each array and is only needed when there are multiple spots of the same gene on the arrays. Varieties are indexed by k but the triplet of indices (i, j, k) for array, dye and variety is redundant. For any given i and j there is only one k. That is, if we know the array and dye, we know which sample it is and thus we know the variety label. We take advantage of the redundant indexing to represent the experimental design in MAANOVA software. The raw intensity data are arranged in columns with a prescribed order, e.g., $(R_1, G_1, R_2, G_2, R_3, G_3, R_4, G_4)$. To specify the design (arrangement of varieties among the arrays) we simply have to indicate which variety labels are associated with the samples in each column. This vector of variety labels, called the *varid*, is used to achieve this specification in MAANOVA.

Estimated Relative Expression

When the ANOVA model is fit to data we obtain estimates for each of the individual terms. Of particular interest in this context are the estimated values of VG. We refer to these as *relative expression values* and denote the estimated values by $\widehat{VG}$. The relative expression value represents the expression level of a gene g in a sample k relative to the weighted average expression of that gene over all of the samples in the experiment. The matrix of $\widehat{VG}$ values has dimensions number of genes $\times$ number of varieties. This derived data combines information across multiple samples of the same variety (and multiple spots of same gene). MAANOVA may be run solely for the purpose of obtaining these derived data. The estimated relative expression values may then be analyzed by other methods, including a second stage analysis of variance or a cluster analysis. The relative expression values are *normalized* data in the sense that effects due to the array, gene, spot, etc., have been removed.

Although the use of relative expression values represents a departure from the customary analysis of ratios, differences in normalized expression values are in fact estimates of the log ratio of the relative expression between two samples (assuming the raw data have been log transformed). Thus it is not a radical departure from the norm. Rather it is a means to achieve a more general interpretation of microarray data. The power of the ANOVA formulation is that it allows investigators to consider experiments that involve more than two samples and to combine information across multiple arrays that are hybridized with experimental samples in (almost) any arrangement. To achieve these goals a concept of relative expression that is more general than pairwise ratios was required.

Mixed Model ANOVA

In the ANOVA model described above, all terms are viewed as fixed but unknown quantities. In an alternative formulation of the ANOVA model, the mixed model, some of the terms are considered to be realizations of a random process. A mixed model formulation for microarray ANOVA in which the spots are treated as random effects has been described by Wolfinger *et al.* (2001). An application of mixed model ANOVA to a complex microarray experiment is described by Jin *et al.* (2001). In its current implementation, MAANOVA will carry out computations for the fixed effects model. Under the mixed model the decomposition of variances is unchanged. However construction of statistical tests and estimators can be different. Future releases of MAANOVA will include functions for mixed model analysis.

2.3 Experimental Design for Microarrays

Microarray experiments are carried out to compare the relative abundance of specific RNA species in two or more biological samples. There may be many samples involved in an experiment and they may have been derived from sources with their own experimental design structure. Jin *et al.* (2001) describe an experiment that involves 8 samples in a fully factorial 2×2×2 arrangement. Other examples include time series experiments (Chu *et al.*, 1998) or treatment versus control comparisons (Callow *et al.*, 2000). It is important that the investigator (and the analyst) understands the structure of the microarray experiment at this level. The distinction between independent biological replicates and technical replicates (obtained from the same biological source) is particularly important for proper construction of test statistics and interpretation of results.

Once the important choices of design at the biological sample level have been made, there is a second layer of design decisions imposed by the paired sample structure of two-color microarrays. A single microarray can only be used to make direct comparisons between two samples. This effectively imposes an incomplete blocking structure on the design, i.e., the samples are paired together on arrays which constitute blocks of size two. Ideally, one might use a balanced incomplete block design in which all possible pairwise comparisons are made directly. However due to the expense of the microarrays or limitations of available sample, this may not be practical. Some solutions to the problem of finding good partial incomplete block design are discussed by Kerr and Churchill (2001a).

Dye–Swap Design

A simple and effective design for the direct comparison of two samples is the dye–swap experiment (figure 1a). This design uses two arrays to compare two samples. On array 1 the control sample is assigned to the red dye and the treatment sample is assigned to the green dye. On array 2 the dye assignments are reversed. This arrangement can be repeated by using 4 (or 6, or more) arrays to compare the same two biological

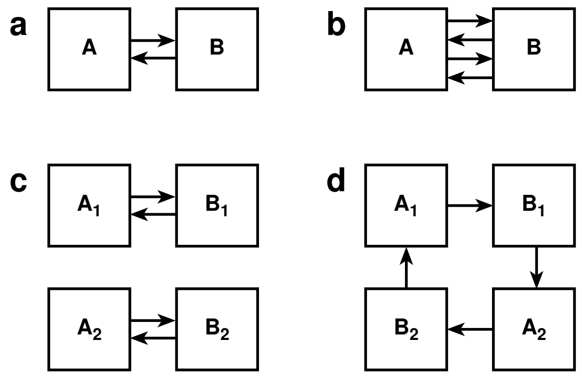

FIGURE 1. Experimental designs for the direct comparison of two samples. Boxes, representing RNA samples, are labeled as varieties A or B. Subscripts indicate independent biological replicates. Arrows represent microarrays. The sample at the tail of the arrow is assayed using the red dye and the sample at the head of the arrow is assayed using the green dye. This figure shows a simple dye–swap (a), a repeated dye–swap (b), a replicated dye–swap (c) and loop design (d).

samples (figure 1b). The repeated dye-swap experiment is useful for reducing technical variation in the measurement but should not be confused with the replicated dye-swap experiment in which independent biological samples are compared (figure 1c). The latter experiment accounts for both technical and biological variation in the assay. It may be more difficult to achieve statistical significance using the replicated dye–swap experiment when biological variation is substantial. However inference in the replicated experiment applies to the biological population from which the samples were obtained. Conclusions from the repeated dye-swap experiment are limited to the samples that were assayed.

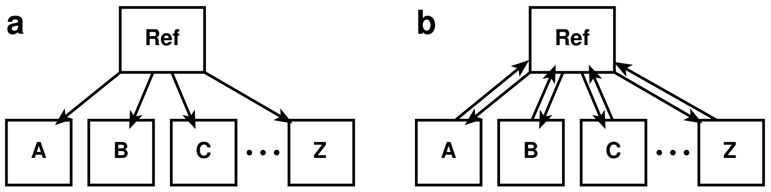

FIGURE 2. Experimental designs employing a reference RNA sample. Boxes represent RNA samples and arrows represent microarrays, as in figure 1. The standard reference design (a) uses a single array to compare each test sample to the reference RNA. A variation (b) utilizes a dye–swap for each comparison.

Reference Design

The classical microarray experiment (*e.g.*, Chu *et al.*, 1998) employs a special RNA sample called the reference sample and all comparisons are made between the test samples and a reference with same direction of dye labeling (figure 2a). In this unbalanced design, the dye by gene (DG) and variety by gene (VG) effects are confounded (Kerr and Churchill 2001a). The MAANOVA software will detect confounding and will drop DG terms from the model. A variation on the classical reference design uses two arrays in a dye–swap configuration to compare each test sample to the reference (figure 2b). This design provides additional technical replicates and eliminates the confounding of relative expression with the gene specific dye effects.

Fully half of the measurements in a reference experiment are made on the reference sample, which is presumably of little or no direct interest. The consequence is that the number of (technical) replicates available for inference is half of what could be achieved using alternative strategies. Despite this inefficiency, reference designs can have a number of advantages. The path connecting any two samples is never longer (or shorter) than two steps. Thus all comparisons are made with equal efficiency. Reference designs can be extended easily (as long the reference sample is available) and can be used to assay large numbers of samples that may have been collected in a (more or less) unplanned fashion.

Loop Designs and More

The loop design (figure 1d), in which samples are compared one to another in a daisy chain fashion, was proposed as an efficient alternative to the reference design by Kerr and Churchill (2001b). In general, small loops provide good average precision. However, depending on the goals of the experiment, large loops may be inefficient. For example, if an investigator wants to compare every pair of samples, loops become inefficient when there are more than ten samples. In addition, the estimation efficiency of a loop is greatly reduced by loss of just a single array, so loops are not a robust design. Variations on loop designs can be achieved by interweaving multiple loops together or by combining loops with reference designs (figure 3).

The possibilities for the design of microarray experiments at the level of arranging pairs of samples onto arrays are perhaps bewildering. However, following a few simple guidelines will ensure that a design is effective for the purposes of a given investigation. Potential biases can be minimized by balancing dyes and samples. Create an even number of technical replicates from each biological sample and assign equal numbers of these to each dye label. It is most efficient to make the comparisons of greatest

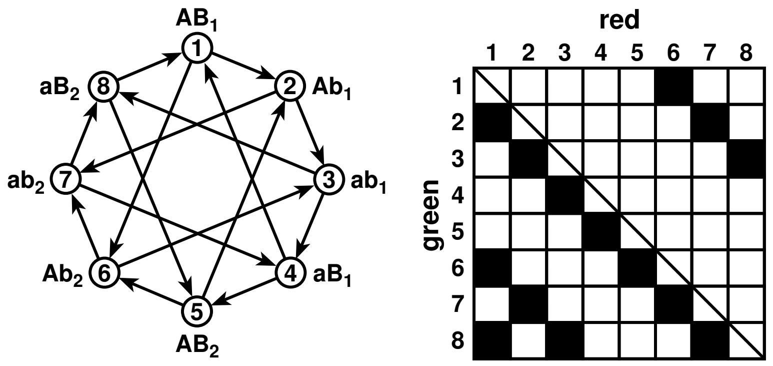

FIGURE 3. A woven loop experimental design. In this experiment there are 8 samples. The diagram on the left is analogous to those in figures 1 and 2. The labels are used to indicate that the experiment has a 2×2 factorial structure with two replicates. The experimental factors are A and B and subscript denote independent biological replicates. On the right is an alternative representation of the same experiment. Each box in the grid represents a possible ordered pairing of samples. Boxes corresponding to pairings used in the experiment are highlighted.

interest directly on the same array. Contrasts between samples that are never directly compared in an experiment are possible provided that there is a path of comparisons linking the two samples. The reference design with dye–swapping is a good design for large experiments because it is simple, robust, and the distance between samples is always two. If conclusions of the analysis will be applied to a biological population, be sure to include independent biological replicates.

Flexibility in the choice of designs has motivated our development of the MAANOVA software. With the confidence of knowing that the complexities of two-color expression assays can be addressed an investigator can focus on the more important aspects of selecting the relevant samples, with adequate biological replication, to address the scientific question of interest. When planning a microarray experiment, it may be helpful to forget for a moment that measurements will obtained on thousands of genes and to design the experiment as if only a single measurement would be obtained on each sample. This perspective can reveal flaws, such as inadequate replication, in a potential design.

2.4 Data Transformations

Before fitting an ANOVA model, the raw data should be transformed to a scale on which the various effects are additive. An argument can be made for the logarithmic scale, however it is commonly observed that the mean and variance of log ratios computed from a single microarray will display systematic features that should be removed prior to analysis.

A standard diagnostic tool for assessing the intensity dependent effects of dyes is the scatter plot of log(R/G) by log(R*G). One plot is generated for each array in an experiment. We refer to these as RI plots (for ratio $\times$ intensity) although there is a precedent for calling them MA plots (Yang et al., 2002). The characteristic curvature seen in RI plots can result from background differences in the two dye channels and/or from differential in the response of the two dyes to laser activation (Cui *et al.*, in prep).

Kerr *et al.* (2002) proposed a shift-log transform to correct the curvature. This method identifies a single constant c that minimizes curvature in the RI plot when values of R and G are replaced by $R + c$ and $G - c$ prior to taking logarithms. The original motivation for this transformation was to shift the raw data to a scale on which a proportional relationship between $\log(R)$ and $\log(G)$ holds, i.e., the symmetric regression line should pass through the origin (Tanner, 1949). In practice shift-log is simple and effective. Yang *et al.* (2001) proposed fitting a smooth curve to the RI plot using the local regression method (lowess) and recentering the log ratio data around the fitted curve. Despite concerns about over fitting we find that the lowess method has advantages and may be applied in cases where the shift-log transform fails to correct the curvature.

Clones on a microarray are printed on regularly spaced grids but the arrangement of clones on the array surface is usually arbitrary. Thus we would not expect to see spatial patterns in the log ratios. We have implemented a version of lowess that simultaneously corrects for spatial and intensity dependent effects. Again, overfitting may be a concern but the spatial lowess function has enabled us to recover some otherwise troublesome data. The assumption underlying all of these data transformations is that the bulk of genes are not differentially expressed. MAANOVA implementations of lowess use robust fitting routines to exclude the influence of outliers that may represent differentially expressed genes.

Another commonly observed feature of the RI plot is the excess variability of log ratios at low intensity. Background subtraction often exaggerates this effect. It has been noted that raw microarray data have both additive and multiplicative error components (Rocke and Durbin, 2001). At high intensity, the multiplicative error dominates and a log transform is appropriate. However at low intensity the signals are small and the additive component of error dominates. A log transform can inflate the variance here. We have proposed a monotone transformation of the raw intensity data, called *linlog*, that behaves like logarithm for high intensity and is linear for low intensity signals to stabilize the variance (Cui *et al.*, in prep). In our experience, setting a transition point that includes about 30% (default) of data points in the linear range tends to stabilize the variance. Application of linlog in combination with either shift–log, lowess or local lowess transformations, produces flat RI plots with stable variance across their entire range. This variance stabilizing transformation should be considered when inference techniques that assume a common error variance across genes will be applied.

Corrections for spatial and intensity dependent effects on the log ratio are essential to avoid being mislead by common artifacts in microarray data. In general it is best to correct biases at the technical level or through clever design (such as dye–swap) rather than rely on post-hoc data adjustments. Simple precautions such as balancing the photo multiplier tube settings when scanning the arrays can be very effective. Correcting biases at the stage of analysis is undesirable because the corrections applied can never be perfectly accurate. It is possible in some cases that attempted corrections may introduce biases greater than the ones they remove. Nonetheless we have found that a small arsenal of data transformation tools is essential for reliable microarray data analysis. Keep in mind that the ANOVA model is an approximation and that transformations are used to improve the quality of this approximation. Our advice is to apply the most gentle transformation that corrects the observed problem.

2.5 Algorithms for computing ANOVA estimates

Estimates of individual terms in the ANOVA model are obtained by the method of least squares. If one makes the assumption that errors (ϵ_{ijkgr} in eqn. 1) are normally distributed, then the least squares estimators are also maximum likelihood estimators. Least squares estimators may be sensitive to outliers in the data and alternative methods of parameter estimation could prove to be more robust. In our experience least squares estimators behave well and we have not implemented robust algorithms in the current version of MAANOVA.

The usual method of fitting ANOVA models by least squares involves calculating the inverse of the *design matrix.* For microarray experiments, this matrix may have dimensions in the tens of thousands and direct inversion is not practical. However we note that the matrix has a regular structure that we can take advantage of to decompose the problem into many smaller calculations. The approach we employ involves fitting the model in two stages. We first fit the *normalization* model

$$Y_{ijkgr} = \mu + A_i + D_j + AD_{ij} \tag{2}$$

to obtain residuals r_{ijkgr}. The rest of the model is fit iteratively on per gene basis. For a each gene g (subscript suppressed) we fit the model

$$r_{ijkr} = G + VG_k + DG_j + AG_i + S_{r(i)} + \epsilon_{ijkr}. \tag{3}$$

In a typical microarray experiment, the same set of genes are assayed on all of the arrays. Thus the factor G is said to be *balanced* with respect to factors A and D and it is a consequence of this balance that the estimates obtained by fitting the model in two stages are identical to those that would be obtained by fitting the whole model (1) is a single step.

Numerically stable algorithms for least squares utilize the QR-decomposition of the inner product of the design matrix (Seber, 1977). In the two stage fitting algorithm the same gene specific model is fit many times. Furthermore randomization tests may require that the whole process be repeated thousands of times. We achieve tremendous computational efficiency by precomputing the QR decomposition of the gene specific design matrix once and storing it.

2.6 Statistical Inference

Parameters obtained by fitting the ANOVA model are estimates and as such are subject to error and uncertainty. In order to ensure that we are not misled into over (or under) interpreting the results, we appeal to methods of statistical inference.

Models

Statistical inference requires the specification of a model for the data. For hypothesis testing inferences it is necessary to specify two models, a null model and an alternative model. As noted above, each RNA sample in an experiment is associated with a label, the variety. Typically RNA samples from a common source will share a common label. However, different labelings of the data can be applied to express different hypotheses about the data. The user of MAANOVA must input a vector of variety identifiers that serve as labels for the samples corresponding to each column of the intensity data. The `varid` is an integer vector and its elements should be chosen from a set of consecutive integers $(1, 2, 3, \ldots)$ with a unique integer for each distinct variety in the model. The structure of varid reflects the design of the experiment. For example a dye–swap is specified as [1 2 2 1], a loop of 5 samples could be [1 2 2 3 3 4 4 5 5 1], a reference design could be [1 6 2 6 3 6 4 6 5 6]. Under the null hypothesis of no differential expression all of the samples would be considered to be one variety (*i.e.*, there are no differences). For the dye–swap and loop experiments, the `varid` is simply a vector of ones, [1 1 1 1] or [1 1 1 1 1 1 1 1 1]. For the reference design, the test samples are considered to be identical but the reference sample is allowed to be distinct. The null model has two varieties and the `varid` would be [1 2 1 2 1 2 1 2 1 2]. A more elaborate example is provided below.

Randomization Methods

Traditionally, statistical inference methods for ANOVA models have appealed either to normality of errors or to large sample theory to establish significance thresholds and/or p-values using tabulated distributions such as χ^2, t, or F. Microarray data often display dramatically non-normal error distributions and samples sizes (on a per gene basis) are usually quite small. Thus we have preferred to use randomization methods, permutation and bootstrapping, to establish distribution free significance levels for statistical tests and confidence intervals. For hypothesis testing applications in MAANOVA, we have implemented permutation methods that shuffle the residuals from fitting a null hypothesis ANOVA model to data. For computing confidence intervals, we employ bootstrapping methods that shuffle residuals obtained under an alternative model. Either procedure may be applied globally or restricted to shuffling within each gene. In bootstrapping applications, an inflation factor is applied to residuals to achieve the correct variance. The shuffling styles available in MAANOVA include restricted or unrestricted shuffling of the model residuals and sample shuffling. Restricted shuffling will shuffle the residuals within genes and unrestricted shuffling will shuffle the residuals globally. Unrestricted shuffling assumes that error terms have a common variance. Restricted shuffling does not require this assumption but it is only practical in large experiments where the number of residuals for each gene is sufficiently large. A third option, sample shuffling, which also requires a large experiment but makes few assumptions, is to shuffle whole arrays. Sample shuffling will freely exchange arrays that have the same (ordered pair of) variety identifiers. Like restricted shuffling it is relatively assumption free but it is ineffective for small experiments. Enhancements to enumerate all possible permutation or bootstrap samples for moderately sized data sets are being considered but the current implementation simply generates a random shuffles. Randomization can be time-consuming but, in light of the dramatic non-normality of microarray data, we consider it to be worth waiting for. The standard tabulated p-values are also available in MAANOVA. The user can expect that the stringency of tests computed under various shuffling options will vary and some judgement is required on a case-by-case basis to make an appropriate choices among these methods.

Hypothesis tests

The MAANOVA package offers three test statistics (called F_1, F_2, and F_3) for hypothesis testing. We routinely compute all three types of tests as each one reveals different aspects of the data. All three test statistics are based on the gene–specific residual sums of squares, denoted by rss_g, and the residual degrees of freedom, denoted by df. Both quantities are model dependent and are available in MAANOVA data objects after a model is fit to the data. Hypothesis testing involves the comparison of two models and test statistics are computed on a gene by gene basis. Thus we can suppress the subscript g and use the notation rss_0, df_0 for null model and rss_1, df_1 for alternative model residual sums of squares and degrees of freedom, respectively.

The statistic F_1 is the usual F statistic that one would compute if data were available for only a single gene,

$$F_1 = \frac{(rss_0 - rss_1)/(df_0 - df_1)}{rss_1/df_1}. \tag{4}$$

It generalizes the t-test approach that is widely used in microarray analysis (Dudoit *et al*, 2002). Significance levels can be established by reference to the standard F distribution or by permutation analysis. This test does not require the assumption of common error variance. However, it has low power in typical microarray experiments because of small sample sizes and it can be sensitive to variations in the estimates of residual variance, rss_1.

The test F_3 explicitly assumes common error variance across all genes. The test statistic is

$$F_3 = \frac{(rss_0 - rss_1)/(df_0 - df_1)}{s^2_{\text{pool}}}, \tag{5}$$

where $s^2_{\text{pool}} = \frac{1}{N}\sum_{g=1}^{N} \text{rss}_{1g}/\text{df}_1$ is the estimated common variance. When testing the null hypothesis of no differential expression, the numerator in eqn. 4 (also in eqns. 5 and 6) is equivalent to $\sum \widehat{VG}^2/df$. The F_3 statistic uses the same denominator for each gene. Thus we are effectively testing based on the magnitude of relative expression values. In the case of two samples, this is equivalent to ranking genes by their log ratio. The F_3 test is powerful and can be applied to small experiments. However it does assume common variance and we recommended checking this assumption, *e.g.* by inspecting residual plots. It is may be necessary to apply a variance stabilizing transformation such as `linlog`.

The test F_2 is a hybrid of the other two tests. The denominator of F_2 uses a gene specific estimate of variance that is shrunken toward the global average variance,

$$F_2 = \frac{(rss_0 - rss_1)/(df_0 - df_1)}{(s^2_{\text{pool}} + rss_1/df_1)/2}. \tag{6}$$

Although it is somewhat ad-hoc in nature, we find that this test performs well in independently replicated experiments (much better than F_1 and slightly better than F_3). The motivating idea was to stabilize the gene specific variance estimates. The approach is similar to SAM t-tests (discussed in Chapter 12) as well to Bayesian approaches employed by Baldi and Long (2001) and by Lönnstedt and Speed (2002). Properties of these "regularized" test statistics is an active area of investigation and it is likely that we will expand the options available in MAANOVA.

Confidence intervals

Formal hypothesis testing for differential expression produces p-values. A p-value for a test of differential expression summarizes the statistical significance of the test statistic, which is based on the variation in gene expression and the error variance. Often times p-values will be considered too concise in that they summarize the *statistical significance* of the data, but they do not give any information to evaluate the *biological significance*. Very small measured differences in gene expression may be statistically significant if the standard error is small, but may be of no interest to a biologist. Confidence intervals allow one to gauge both the statistical significance and the potential biological significance of the result by providing the precision and magnitude of the changes in relative expression.

Confidence intervals are computed via bootstrapping to avoid normality assumptions. Currently two kinds of confidence intervals can be computed. One kind is based on the assumption of a global error variance, analogous to the F-test F3, and produces confidence intervals of uniform width for all genes. A second kind is based on the assumption of a gene-specific error-variance, analogous to the F-test F1. Confidence interval methods are another area of active investigation (see, for example, Kerr *et al.*, 2002) where we anticipate advances that will be incorporated in MAANOVA.

Multiple test adjustment

When we compute test statistics or confidence intervals for differential expression we are simultaneously conducting thousands of inferences — one for each gene on our arrays. A well-recognized problem of this multiplicity is that the chances of obtaining a positive result becomes high, even if all null hypotheses are true. The most common approach to the multiple-testing problem is to control the *family-wise error rate.* The tests are done at a level of stringency so that the probability of making one or more type I errors is smaller than some nominal *alpha* level. Many scientists find this kind of control to be overly conservative for microarray studies. If the goal of an experiment is to generate a list of interesting genes, a certain number of false positive results may be tolerable.

As alternative approach to multiple test adjustment is with *false discovery rates* (FDR, Benjamin and Hochberg, 1995). The false discovery rate is defined to be the expected proportion of false positives among all rejected hypotheses. Tusher, Tibshirani, and Chu (2001) incorporate a method for estimating the FDR in the SAM methodology for microarrays. However, FDR estimation is separable from the other aspects of the SAM methodology. The major requirement of the general methodology is that the null versions of the test statistics can be simulated. In the context of ANOVA methodology for microarrays, null test statistics can be simulated by permuting the data to preserve the experimental design except for the variety identifiers. Obviously, a certain amount of replication in the experimental design is required for this to be effective. For example, if the design is a loop with 4 slides then there are only 4!=24 permutations of the array data, but many of these will be equivalent relative to the F-statistic. There are really only 24/4=6 permutations, hardly sufficient to simulate a distribution.

The current implementations of F tests and confidence intervals in MAANOVA include the one-step adjustment method of Westfall and Young (1993) to control familywise error rates. The next release will include an implementation of the Westfall and Young step-down method for adjusted p-values and methods for estimating false discovery rates.

2.7 Cluster Analysis

The estimated relative expression values ($\widehat{VG}$) obtained from fitting the ANOVA model capture the “profile” of expression across the samples in an experiment. We can use these quantities to cluster the genes or samples. By “clustering” we mean organizing the genes or samples into a hierarchical or grouped structure that represents the degree of similarity among profile. The clustering structure is denoted by C. Prior to clustering the user may want to filter out the insignificant genes using F test results. The current implementation of MAANOVA includes a variety of options for hierarchical clustering as well a k-means function. These function are useful for organization and interpretation of long lists of significant genes.

Bootstrap clustering

Our approach to clustering with expression data involves fitting the ANOVA model to obtain a derived data set of relative expression values. A standard clustering algorithm is applied to these. Schematically, the process is $y \rightarrow VG \rightarrow C$. A cluster analysis will always produce a clustering C but often there is no indication of how reliable it is.

The MAANOVA software includes features to assess the reliability of C by bootstrapping (Kerr and Churchill 2001c). The steps are:

1. generate bootstrap data set $y^\star$ by residual shuffling,
2. fit the ANOVA model to $y^\star$ to obtain $VG^\star$,
3. generate a clustering $C^\star$ from the $VG^\star$ matrix,
4. repeat steps 1–4 N times,

This process will yield N clusterings $C^\star$ and it will be necessary to obtain a summary. MAANOVA include functions to generate and report the stable sets, i.e., sets of genes (or samples) that are grouped together in at least proportion p of all clusterings $C^\star$. For cluster analysis methods that partition a set of objects into a fixed number of groups, the stable sets are trivial to summarize. For hierarchical clustering, we use a consensus tree approach.

Consensus tree methods are widely used phylogenetic research to summarize the common features among a set of trees (Felsenstein, 1985). Consensus trees are typically multifurcating, whereas the trees being summarized may be all bifurcating. Collapsing of the bifurcating structure in the consensus trees indicates the absence of a consistent structure in the set of trees being summarized. The different types of consensus trees and algorithms for constructing them are described by Margush and McMorris (1981).

A clade is a grouping of objects defined by a tree. For each branch in a tree, there is a unique clade consisting of all tips below the branch. A majority rule consensus tree requires that a clade is included if and only if the same clade occurs in at least half of the trees. A generalization of the majority rule consensus tree requires the clade occur in a proportion p of the trees, where $p > 1/2$ to ensure that self-contradictory clades do not appear in the consensus tree. The algorithm for consensus tree building used in our software is quick and straightforward. We store all the clades for all the trees in the bootstrap set, count the total occurrence of each and then construct a list of clades in the consensus tree.

3 Software

3.1 Availability

MAANOVA functions have been developed and tested in Matlab Release 12 for Windows and Linux Redhat 7.0. Some functions are written in C for speed. Executable software and source code can be downloaded from http://www.jax.org/research/churchill/software/anova/. The MAANOVA functions can be called from within the Matlab environment as part of an interactive data analysis session. Alternatively, they may be run as scripts or incorporated in other user defined Matlab functions. The user has access to standard Matlab functions and complete freedom to manipulate data objects in the analysis environment. We find this kind of freedom and flexibility to be appealing and preferable to compiled applications which limit the analyst control over data objects. However, with freedom comes a certain degree of responsibility for knowing what you are doing. Some example datasets and scripted analyses are provided at the web site listed above.

This section outlines the primary functions of MAANOVA in groups according to their standard usage. MAANOVA functions are designed to operate on special data structures (objects) and we provide descriptions of these here. For detailed information on function syntax, the user can type "`help [functionname]`" in the Matlab environment. The fields contained in any data object can be listed by simply typing the object name. A users' manual is in preparation and will be regularly updated to reflect changes in the code. Lastly, the source code of each function is available and can be consulted or modified as needed by the user.

In parallel with our development of the Matlab version of MAANOVA we have created an implementation of MAANOVA in the R programming environment. There are minor differences in syntax (*e.g.*, underscore is a reserved character in R) but the functions and data structures are essentially identical. The R environment is freely available and a number of other microarray analysis packages are being developed in R (see chapter 1). We will maintain the R version and we plan to provide functions for creating and converting data objects to facilitate interoperability with SMA and other R-based packages.

3.2 Functionality

The MAANOVA package continues to grow. Rather than list all of the available function here, we will highlight some of the most useful or important functions. A great strength of MAANOVA, in both the Matlab and R versions, is the capability for the user to define her own functions and carry out analyses that could not be anticipated by the developers. Details regarding the syntax, and input and output data structures for MAANOVA commands are available through help functions in both Matlab and R environments.

Importing Data

Data can be read into the Matlab environment using one of several built in functions such as `load`, `tblread`, `dlmread`, or `textread`. The raw intensity data should be formatted as a numerical data matrix with alternating columns representing the untransformed R and G values. Each array is represented by an adjacent pair of columns. Rows of the data matrix correspond to spots on the array and if there are duplicated spots these should be arranged in adjacent rows and should be identical in number for each spot. It is not possible to work with unequal number of duplicates in the MAANOVA model and in some cases data may have to be discarded or excess replicates treated as separate genes. The arrangement of data is best done outside of the Matlab environment using a spreadsheet application. In addition to the raw intensity data, the user may wish input gene identifiers and information about the array layout, *e.g.*, meta–row, meta-column, row, and column positions for each spot.

Creating the data and model objects

After reading in the raw data, the user can create several important objects. The function `createData` is used to create the data objects and function `makeModel` is used to create model objects. The data and model objects contain all of the information of the experiment design and are required inputs for most MAANOVA functions. The data object and the model object are created and maintained separately to allow the user to apply more than one model to a given data set.

The function `createData` takes raw intensity data and the number of replicates as command line inputs and creates a data object with these fields:

`narrays`: total number of arrays in the experiment

`ngenes`: total number of genes in the experiment `nspots`: number spots for each gene in total across all arrays

`nreps`: the number of replicates of each gene on one array

`data`: the raw intensity data

`adjdata`: transformed data

`colmeans`: column means for the transformed data

`offset`: offset values from `shift`, `lowess`, or `linlogShift` function

`method`: a string to indicate the data transformation method used.

By default, the `adjdata` field contains log transformed data. This can be changed by calling one of the data transformation functions. The type of transformation applied and auxiliary information are stored in the fields `method` and `offset`.

The `makeModel` function takes a microarray data object, a `varid` vector and a model indicator as input. The model indicator is an integer array with three elements to indicate whether to fit VG, DG and AG effects or not. For example, if a user wants to fit only VG and AG effect and leave out DG effect, this variable should be [1 0 1]. The model object has fields:

`[fitVG fitDG fitAG]`: flags to include terms in model

`varid`: vector of variety identifiers

`nvars`: total number of distinct varieties in the model `varcount`: counts of the occurrence of each variety in varid

`even_flag`: indicates that the design is even.

`latsq_flag`: indicates that the design is a latin square

`VDCon_flag`: indicates that terms V and D are completely confounded

`X`: the design matrix for a single gene in the experiment

`A`: contrasts for zero-sum constraints on the design matrix

`Q, R`: QR decomposition of the design matrix

The model object stores several derived quantities that are used to speed other computations and is not advisable to modify this object once it is created.

Data Transformation and Visualization Functions

The first steps of an analysis often involve getting the data onto the right scale. An example of visualization tools used in conjuction with a data transformation is shown in figure 4. MAANOVA provides a variety of functions for data transformations including `shift`, `lowess`, and `linlog`. The `lowess` function includes options for spatial smoothing and for linear or quadratic fits. Results of applying a transformation function are stored in the `adjdata` field of the data object. The method of transformation is recorded along with auxiliary information such as offsets for shift–log transform. A data object can only store one type of transformed data. If more than one transformation is required, the user should create multiple data objects.

The function `riplot` takes a data object as input and generates a set of scatterplots of log ratio by log total intensity. The user can highlight selected subsets of genes which can useful for diagnosis of problems with data or simply for viewing interesting subsets of data. The function `arrayview` can be used to display any function of the data as a color scale on the grid coordinates of an array. Used in conjunction with the function `make_Ratio` it can be used to assess spatial heterogeneity in log ratios. Matlab and R environments provide powerful and flexible graphical functions that can be used to view data. Some examples are provided in our online scripts. The user is encouraged to explore the possibilities and be creative.

Model fitting and diagnostics

The least squares fit of an ANOVA model to a dataset is achieved by a call to the function `fitmaanova` which takes a data object and a model object as arguments. There is an optional flag to suppress computation of the sums of squares for the ANOVA table. This can be useful when things get slow. The output of `fitmaanova` is an ANOVA object with fields:

`yhat`: matrix of fitted value

`rss`: residual sum of squares for each gene

`G`: estimated gene effects

`VG`: estimated relative expression values

`DG`: estimated gene specific dye effects

`AG`: estimated spot effects

`model` : a vector of flags `[fitVG, fitDG, fitAG]`that indicate which model terms were fit to the data.

`table`: cell array containing the ANOVA table

Fitting of the ANOVA model is the core function of our software package. It is written in C and may be called repeatedly by other analysis functions.

After fitting the ANOVA model, it is recommended to check results by generating graphical diagnostics. The function `resiplot` will generate a standard residual plot. A variety of different scatter plots can be useful and examples are provided in our online scripts. It most effective to use built in graphics functions for constructing these plots. The `arrayview` function can be used to visualize the spatial patterns of VG, AG, DG and residuals.

Detecting Differentially Expressed Genes

The function `make_Ftest` will compute test statistics and carry out a permutation analysis of their distribution. It takes a data object and two model objects (a null model and an alternative model) as arguments. In addition, the user can set the number of permutations, the significance level(s), the type of shuffling, and the type(s) of F test method to use. The output of `make_Ftest` is an object with fields:

`SigLevel`: user specified significant levels

`nIteration`: number of permutations

`shuffle_flag`: indicates the shuffling method used

`method`: type of F test computed, may be any combination of [123]

`NullModel`: ANOVA terms fit under the null model

`AltModel`: ANOVA terms fit under the alternative model

`NullVarid`: null model variety identifiers

`AltVarid`: alternative model variety identifiers

`F1, F2, F3`: structure arrays for F test result. Each contains the following fields:

> `Fobs`: observed F values
>
> `Fcritpg`: calculated F critical value per gene
>
> `Pvalpg`: calculated permutation P value per gene
>
> `Fcritmax`: multiple test adjusted F critical value
>
> `Pvalmax`: multiple test adjusted P value
>
> `Ptab`: tabulated p value from F distribution (F1 only).

A 'volcano' plot provides a graphical summary of the simultaneous results from all three F-tests. The function `volcano` takes an anova object, an F–test object and significance thresholds for each test as input arguments. The y axis of the volcano plot is the $-\log_{10}$(tabulated p-value) for the F1 test. If the experiment has only two samples, the x axis is shown as the difference in estimated relative expression values,

$\log_2(\widehat{VG}_{1g} - \widehat{VG}_{2g})$. The resulting figure has the appearance of an erupting volcano. When there are more than two samples, the x axis is the root mean square of the relative expression values, $\sqrt(\sum_{i=1}^{k}(\widehat{VG}_{kg})^2)$. Although the volcano–like appearance is lost in the modified version, we kept the catchy name. A horizontal line on the plot represents the significance threshold of the F1 test. Vertical lines represent thresholds for the F3 test and red color is used to indicate genes selected by F2 test. Genes in the upper-right (and upper-left) corner(s) of the figure are significant by all three criteria and their indices are returned by the `volcano` function.

Confidence intervals

The function `make_CI` is used to construct confidence intervals for user defined contrasts of the relative expression values. Input arguments are a data object, a model object, a set of contrasts, and parameters to specify significance levels and the mode of shuffling. The default set of contrasts is all pairwise differences, i.e. all possible log ratios. Bootstrapping can be carried out at the individual gene level or with a multiple testing adjustment. The output of this function is an object with fields:

`shuffle_flag`: shuffle method:

`output_selection`: indicates type of multiple test adjustment

`SigLevel`: significance level

`Contrast`: contrast set

`nIterations`: number of bootstrapping iterations

`CI`: confidence interval for each gene

`CImaxLo, CImaxHi`: limits for adjusted confidence intervals

Cluster analysis

Cluster analysis can be carried out using built in functions in Matlab and R. In Matlab, we call built in functions `pdist` and `cluster` from the Statistics Toolbox. In R we call finctions `dist`, `hclust` and `kmeans` from the `mva` package. The MAANOVA function `boothc` will run bootstrap analysis on a hierarchical clustering. It takes a data object, a model object, the index of the genes to be used in clustering, a flag for clustering genes or samples, and the number of bootstrap iterations as input arguments. It returns an object, `nodeobj`, that lists all of the clades in the bootstrap set and their frequencies. The function `consensus` operates on the node object to create a consensus tree. It call the `drawconsensus` function to draw the tree and outputs an object to represent the consensus tree. The function `writephylip` can output the consensus tree to a text file in PHYLIP format (http://evolution.genetics.washington.edu/phylip.html) for use with other tree drawing software.

The MAANOVA package includes a function `kmeans` to compute K-means cluster analysis of (selected) genes from their relative expression values. The function `fom` is useful for determining the number of groups for K-means analysis and the function `bootkmean` will run a bootstrapping analysis. The results of bootstrapping a K-means cluster analysis can be visualized with the `VGprofile` function.

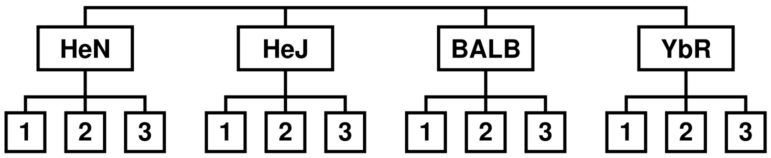

FIGURE 5. Experimental design for tumor survey data. Three independent tumor samples were obtained from each of 4 strains of mice. There are 12 independent RNA samples in total. The arrangement is a standard one–way layout with three replicates per group. The tumor samples were assayed with 24 microarrays using the dye–swap reference design as shown in figure 2b.

4 Data analysis with MAANOVA

In this section, we will demonstrate data analyses using MAANOVA. The illustration is brief and is intended to highlight unique or commonly used features of MAANOVA. A complete script and data for this analysis are available on our web site.

The design of experiment at the level of the biological samples is shown in figure 5. Three independent mammary tumor samples were obtained from each of 4 different strains of mice. These samples were compared using 24 microarrays. Each test sample was compared to a reference RNA using a dye–swap arrangement as in figure 2b. Each microarray contained approximately 30,000 spots representing 15,000 genes printed in duplicate.

The raw intensity were read into the Matlab environment and named `pmt` and there are 2 duplicated spots per gene. The row and column locations of each spot are stored in vectors `grow` and `gcol`. Function calls

```
> data = createData(pmt,2);

> data = malowess(data, grow, gcol, 'lowess2');
```

were used to create a data object and to apply the spatial lowess transformation.

We consider three models for the data. Model 0 assumes that all samples are the same, *i.e.,* no differential expression of genes. Model 1 assumes that samples are the same within a strain but may vary between strains. Model 2 allows each sample to be unique. We first specify the variety identifiers.

```
> varid0 = [2 1 1 2 1 2 2 1 2 1 1 2 1 2 2 1 2 1 1 2 2 1 1 2 ...
2 1 1 2 1 2 2 1 2 1 1 2 2 1 1 2 2 1 1 2 2 1 1 2];
> varid1 = [5 1 1 5 1 5 5 1 5 1 1 5 2 5 5 2 5 2 2 5 5 2 2 5 ...
5 3 3 5 3 5 5 3 5 3 3 5 5 4 4 5 5 4 4 5 5 4 4 5];
> varid2 = [13 1 1 13 2 13 13 2 13 3 3 13 4 13 13 4 13 5 5 13 13 6 ...
6 13 13 7 7 13 8 13 13 8 13 9 9 13 13 10 10 13 13 11 11 13 13 12 12 13];
```

Then we fit the ANOVA models.

```
> model0 = makeModel(data,varid0,[1 1 1]);
> anova0 = fitmaanova(data,model0);
```

The argument "[1 1 1]" in `makeModel` indicates that all of the terms VG, DG and AG are included in the model. Similar commands with `varid1` and `varid2` in place of `varid0` are issued to fit Models 1 and 2.

We computed F tests to compare the various models. To compare models 0 and 1, issue the command

```
> Ftest0 = make_Ftest(data, model0,model1,0.95,500);
```

The last two arguments specify the confidence level and the number of permutations, respectively. The results of this F test analysis are summarized as a volcano plot in figure 6. Twenty genes were identified as being significant. The function call to generate the volcano plot is

```
> idx0= volcano(anova1,Ftest0, 0.001, 0.05, 0.05);
```

Where the last three arguments are significance levels for tests F1, F2 and F3 respectively.

The 20 significant genes identified by `Ftest0` were clustered using the K-means algorithm. The function call to generate the kmeans analysis is

```
> [class,grp0] = bootkmean(data, model2, idx0, (1:12), 10, 500, 0.8, ...
'gene', 'VG');
```

Selected genes are indexed by `idx0`, selected samples are 1 through 12, the number of groups to fit is 10, the number of shuffles is 500, and results will be accepted at 0.8 confidence. The argument ‘gene’ specifies that we are clustering genes not samples and ‘VG’ specifies sample shuffling option. The function call

```
> VGprofile(VGdiff,grp0)
```

will generate the clustered relative expression profiles shown in Figure 7. Note that the testing and selection of genes was based on model 1 but the cluster analysis uses the unrestricted estimates of relative expression in model 2.

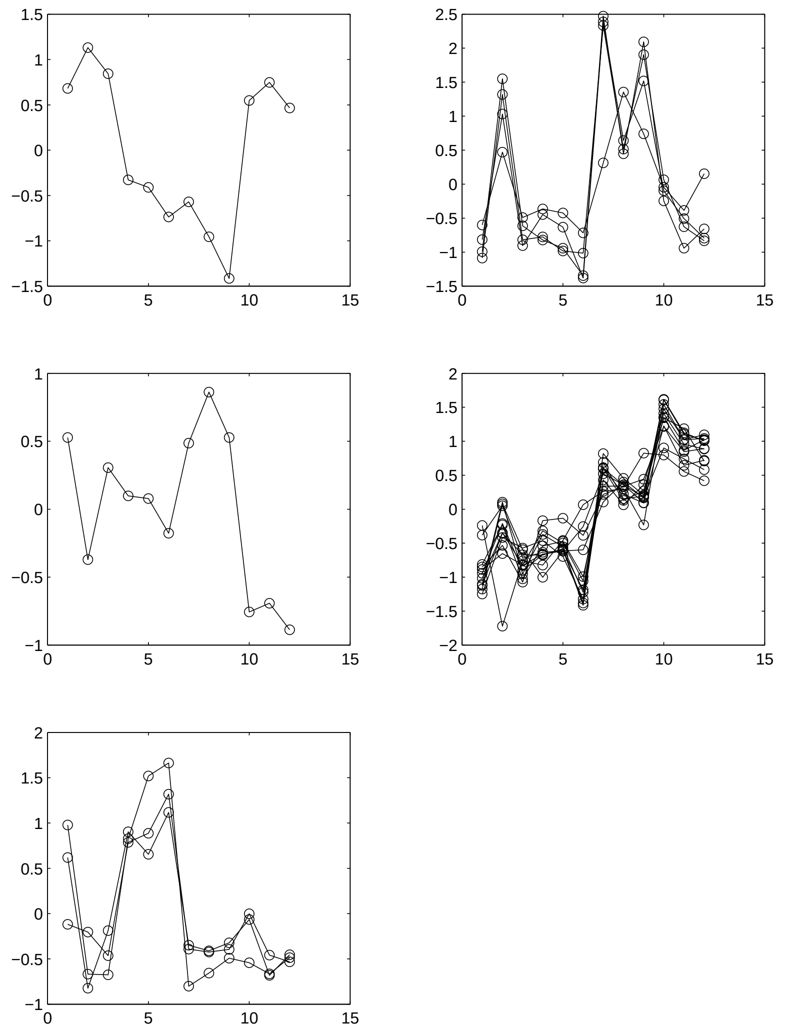

FIGURE 7. A K-means analysis of expression profiles from the tumor survey. Profiles are shown for the 20 significant genes selected by F-tests using the tumor survey data (figure 6). The profiles cluster into 5 groups and all of the assignments are supported at the 80% level by bootstrap analysis. In each panel, the x-axis is the sample index and the y-axis is the relative expression value.

An objective of this study was to classify the samples according to their observed pattern of expression. For this purpose we identified genes that were differently expressed without regard to strain of the sample by conducting a test of model 2 compared to model 0. Indices of significant genes according to this test were stored in `idx2` and a hierarchical cluster analysis of the samples was done. The function call

```
> nodeobj = boothc(data, model2, idx2, (1:12),500,'seuclid', ...
'sample', 'VG');
```

will generate 500 bootstrap trees using the standardized euclidean metric to compute the distance. The last two arguments indicate that we are clustering the samples and using the sample shuffling method.

The function call

```
> ctobj = consensus(nodeobj);
```

will generate the cluster diagram shown in Figure 8. The hierarchical cluster analysis was assessed using a sample bootstrap and support values are shown on the tree. It is interesting to note that the clusters are largely concordant with strain but there is a outlier that is strongly supported by the bootstrap analysis.

5 Discussion

There are several directions in which we plan to further the development of MAANOVA. Of these, the most substantial is the extension to mixed effects ANOVA. As a first step we plan to mimic the approach of Wolfinger *et al.* (2001). We are investigating alterna-

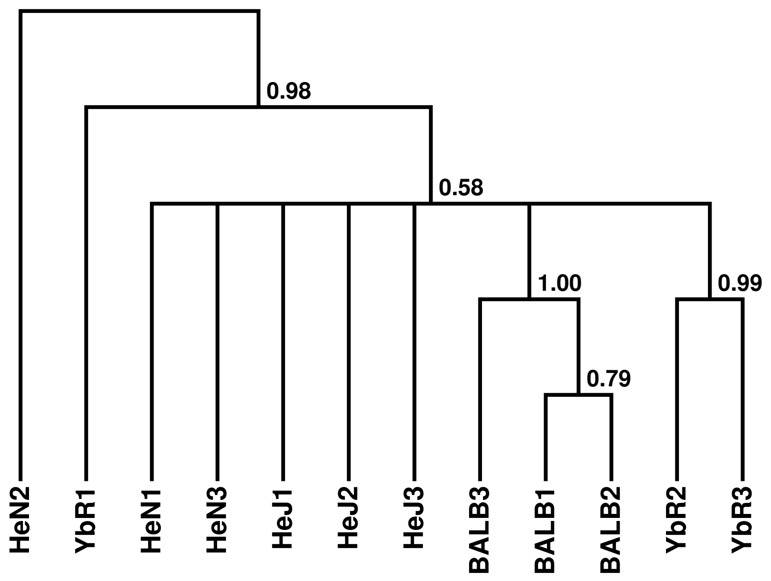

FIGURE 8. A majority rule consensus tree of tumor samples. A consensus was constructed using 500 bootstrap samples (residual method) from the tumor survey data. Strain and sample number of the tumors are indicated on the tips of the tree. Numbers on the branches indicate the proportion of bootstrap samples that support the clade, the grouping of sampled below each branch. Branches that are supported at less than 50% are collapsed to a multifurcation in the consensus tree. The pattern of clustering is largely consistent with the strain origins of the tumors. A notable exception is HeN2 which appears to be a novel variant.

tive decompositions of the sums-of squares in order to expand the scope of options for randomization tests in MAANOVA and we are developing a more general approach to model specification in order to address hypothesis testing in a mixed model context. In a complex experiment where the RNA samples have a non-trivial experiment design structure, it may be desirable to include both fixed and random terms in the VG component of the model. Mixed models can admit more general covariance structure and will provide shrinkage of estimated effects that can reduce bias.

Inevitably there are missing data in microarray experiments. Missing data occur for any number of reasons but are often due to a technical failure of some spots on the array due to dust, scratches or printing error. The algorithm employed to fit ANOVA models in two stages requires that the same set of genes be present on all arrays. Thus if a gene is missing on just one array (or a few) array(s) in a large experiment, it must be removed from the whole. In some case when modest number of genes are missing at random from a large number of arrays, a substantial portion of the whole experiment may have to be discarded. This is clearly undesirable. The development of missing data techniques that would allow us to retain useful information on genes with partial data would be desirable.

It seems likely that additional dyes will be developed for use with microarrays. Already there are scanners on the market that can accommodate as many as five wavelengths. The basic structure of the ANOVA model is not affected by the use of multiple dye labels on the same array. The use of multiple dye promises to substantially improve the accuracy of microarray experiments by effectively increasing the *block size* from two to three or more.

MAANOVA was originally developed for in house use as a development and testing platform for microarray analysis. For these purposes, the interactive environments provided by Matlab or R are ideal. We use the same environment to carry out data analysis tasks. However, not everyone is comfortable with command driven environments and a different interface may be useful. Toward this end we are actively discussing the form and function of a graphical user interface for MAANOVA.

Literature Cited

Baldi P, Long AD (2001) A Bayesian framework for the analysis of microarray expression data: regularized t-test and statistical inferences of gene changes. *Bioinformatics* 17: 509

Benjamin Y, Hochberg Y (1995) Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J Royal Statistical Society*, Series B 57: 289

Callow MJ, Dudoit S, Gong EL, Speed TP, Rubin E (2000) Microarray expression profiling identifies genes with altered expression in HDL deficient mice. *Genome Research* 10: 2022

Chu S, DeRisi J, Eisen M, Mullholland J, Botstein D, Brown PO (1998) *Science* 282: 699

Cui XQ, Kerr MK, Churchill GA (in prep) Data transformations for normalization of cDNA microarray data.

Felsenstein J (1985) Confidence limits on phylogenies – An approach using the bootstrap. *Evolution* 39:783

Jin W, Riley RM, Wolfinger RD, White KP, Passador-Gurgel G, Gibson G (2001) The contribution of sex, genotype and age to transcriptional variance. in Drosophila melanogaster. *Nature Genetics* 29:389

Kerr MK, Afshari CA, Bennett L, Bushel P, Martinez J, Walker N, Churchill GA

Statistical analysis of a gene expression microarray experiment with replication. *Statistica Sinica* 12:203

Kerr, M.K., Churchill, G.A. (2001a) Experimental design for gene expression microarrays. *Biostatistics* 2:183

Kerr, M.K., Churchill, G.A. (2001b) Statistical design and the analysis of gene expression microarray data. *Genetical Research* 77:123

Kerr MK and Churchill GA (2001c) Bootstrapping cluster analysis: Assessing the reliability of conclusion from microarray experiments. *PNAS* 98:8961

Kerr, M.K., Martin, M., Churchill, G.A. (2000) Analysis of variance for gene expression microarray data. *J Computational Biology* 7:819

Kerr MK, Leiter EH, Picard L, Churchill GA (2002) Sources of Variation in Microarray Experiments. In: Computational and Statistical Approaches to Genomics, edited by Wei Zhang and Ilya Shmulevich Kluwer Academic Publishers, pp 41

Lönnstedt, I and Speed, TP (2002) Replicated microarray data. *Statistica Sinica* 12: 31

Margush, T. and F. R. McMorris (1981) Consensus n-trees. *Bulletin of Mathematical Biology* 43: 239

Pritchard CC, Hsu L, Delrow J, Nelson PS (2001) Project normal: Defining normal variation in mouse gene expression *PNAS* 98:13266

Rocke DM and Durbin B (2001) A model for measurement error for gene expression arrays. *J Comp Biol* 8: 557

Seber, GAF (1977) "Linear Regression Analysis". Wiley, NY.

Schena, M (ed.) (2000) *DNA microarrays: A practical approach* (Practical Approach Series, 205). Oxford University Press, Oxford.

Tanner JM (1949) Fallacy of per-weight and per-surface area standards, and their relation to spurious correlations. *J Appl Physiol* 2:1

Tusher VG, Tibshirani R, Chu G (2001) Significance analysis of microarrays applied to the ionizing radiation response. *PNAS* 98: 5116

Wolfinger RD, Gibson G, Wolfinger ED, Bennett L, Hamadeh H, Bushel P, Ashfari C, Paules RS (2001) Assessing gene significance from cDNA microarray expression data via mixed models. *J Comp Biol* 8:625

Yang YH, Buckley MJ, Dudoit S, Speed TP (2002) Comparison of methods for image analysis on cDNA microarray data. *J Comp Graph Stat* 11: 108

Yang YH, Dudoit S, Luu P, Lin DM Peng V, Ngai J, Speed TP. (2001) Normalizatin for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation. *Nucleic Acids Research* 2002 30:e15.