

ANALYSIS AND QUALITY CONTROL OF cDNA MICROARRAY DATA

PREPARED BY MATTHEW J. RODLAND

DEPT OF MEDICAL INFORMATICS AND OUTCOMES RESEARCH

OREGON HEALTH & SCIENCE UNIVERSITY,
DECEMBER 13, 2001

School of Medicine
Oregon Health & Science University

CERTIFICATE OF APPROVAL

This is to certify that the Masters thesis of

Matthew J. Rodland

has been approved

[Redacted Signature]

Professor in charge of thesis

[Redacted Signature]

Member

[Redacted Signature]

Member

2/14/01

ANALYSIS AND QUALITY CONTROL OF cDNA MICROARRAY DATA

DEPT OF MEDICAL INFORMATICS AND OUTCOMES RESEARCH

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	6
INTRODUCTION.....	7
Abstract.....	7
Background.....	7
Introduction to Microarray Analysis.....	8
Research Question.....	10
Figure 1: Microarray Process.....	12
Sources of Variation by Phase.....	13
RNA and Probe Preparation.....	13
Target Spot and Print Quality.....	13
Hybridization.....	13
Image Scanning.....	14
Software Analysis.....	14
Summary.....	14
Introduction to Data Processing and Normalization Strategies.....	14
Yang, Y.H. Et al. Normalization for cDNA Microarray Data, SPIE Bios[6].....	15
Schuchhardt, J. et al. Normalization Strategies for cDNA Microarrays, Nucleic Acids Research [5].....	15
Newton, M.A. et al. On Differential Variability of Expression Ratios: Improving Statistical Inference about Gene Expression Changes from Microarray Data, Journal of Computational Biology [7].....	16
Richmond, C.S. et al. Genome-wide Expression Profiling in Escherichia coli K-12, Nucleic Acids Research [9].....	17

ANALYSIS AND QUALITY CONTROL OF cDNA MICROARRAY DATA

DEPT OF MEDICAL INFORMATICS AND OUTCOMES RESEARCH

Delenstarr, G. et al. Estimation of Confidence Limits of Oligonucleotide Array-based Measurements of Differential Expression, Proceedings of SPIE [10].....	17
Summary.....	18
METHODS.....	19
Quality Assurance.....	19
Data Processing and Normalization.....	19
Quality Control.....	22
Reporting and Experimental Design.....	23
Summary.....	24
RESULTS.....	25
Quality Assurance.....	25
Comparison of Normalization Methods.....	25
Standards or "housekeeping" genes.....	25
Figure 2: "Good" Standards Duplicates.....	26
20% Trimmed Mean (Slidewise).....	27
Figure 3: 20% Slidewise Trimmed Mean Duplicates.....	28
Gridwise.....	28
Figure 4: Gridwise vs. Slidewise.....	30
Log Ratio.....	30
Figure 5: ArrayStat Vs Fold Change (Excel).....	32
Figure 6: ArrayStat vs. Fold Change.....	33
Summary.....	33
Quality Control.....	33
Reporting and Experimental Design.....	35

ANALYSIS AND QUALITY CONTROL OF cDNA MICROARRAY DATA

DEPT OF MEDICAL INFORMATICS AND OUTCOMES RESEARCH

Summary.....	35
CONCLUSIONS AND FUTURE DIRECTIONS.....	37
BIBLIOGRAPHY.....	38
APPENDIX A – DATA DICTIONARY.....	39
APPENDIX B – MICROARRAY PREPARATION PROTOCOLS.....	47
Indirect, Biotinylated cDNA synthesis.....	50
Direct Labeling, cDNA synthesis (label directly incorporated into cDNA).....	52
PREPARATION OF SLIDES FOR SCANNING.....	55
APPENDIX C – AUTOMATED NORMALIZATION PROTOCOL.....	65
Section 0: Preparation.....	67
Section 1: Import Raw Text Data from BARRY Oracle Data Warehouse to Excel.....	68
Section 2: Local correction for background (mean signal – median background).....	70
Section 3: Normalize Excel Data.....	71
20% Trimmean all genes by channel / slide.....	71
Meta Grid Normalization.....	72
Section 4: Analyze Duplicates.....	73
Section 5: Calculate Ratios (by slide).....	77
Section 6: Add Gene Name to Accession Number.....	80

ANALYSIS AND QUALITY CONTROL OF cDNA MICROARRAY DATA

DEPT OF MEDICAL INFORMATICS AND OUTCOMES RESEARCH

Section 7: Sort Results by Accession Number.....80

Section 8: Filter Ratios by Fold Change.....80

APPENDIX D – DOCUMENTED AUTOMATED NORMALIZATION PROGRAM
SOURCE CODE.....82

ACKNOWLEDGEMENTS

Mark Turner – Assistance with program coding, program for exporting normalized data to GenExplore™, Oracle database and SQL coding support, helpful input on implementation of data processing, normalization, and quality control schemes

Brian Olmstead – Creator of BArray Microarray Data Warehouse for importing raw microarray data from AutoGene

Srinivasa Nagalla, M.D. – Assistant Professor and Director of the cDNA Microarray Core Facility, Dept. of Pediatrics, Oregon Health & Sciences University, Thesis Committee

Christopher Dubay, Ph.D. - Assistant Professor, Dept. of Medical Informatics and Outcomes Research, Thesis Advisor

Motomi Mori, Ph.D. - Associate Professor and Director of Biostatistics, Thesis Committee, provided input on statistical methods used in data processing and normalization

Daniel Baker, M.D. - Creator of external program to inspect original images for spot quality

James "Storm" T. Shirley – Suggested method for reporting grid alignment quality

INTRODUCTION

ABSTRACT

Analyzing and reporting data generated from cDNA microarray experiments is a challenging process. There are many potential sources of error that must be controlled to obtain valid and reproducible results from these experiments and estimate relative levels of gene expression. Advanced normalization and data processing of the raw microarray data are required for the exchange and quality assurance of gene expression results. Normalization methods are typically used to correct for systematic sources of variation in cDNA microarray experiments. Data processing refers to all other methods used in microarray analysis, e.g. background subtraction. There is currently no proven best method of normalization or data processing for the analysis of microarray data. We wanted to find out whether we could generate a consistent and meaningful report of microarray gene expression results, correcting for systematic sources of variation, and ensure that the results are valid and reproducible for the genetics researcher. To answer this question, we compared several methods in our lab in an attempt to determine which is the most valid and reproducible for our experiments. While the methods for data processing might vary depending on the structure and design of the individual experiment, we found that overall our most reliable method for normalization was to calculate a 20% trimmed mean of all spots on the microarray, and then divide the local corrected value for each spot on the array by the trimmed mean. The five phases of the microarray process: RNA and probe preparation, target spot and print quality, hybridization, image scanning, and software analysis are shown in Figure 1. We produced several measures of overall slide intensity for quality control, as well as writing and publishing protocols on the Internet for the microarray process up to the image scanning and software analysis phase. To automate the data processing and normalization of microarray data and report the results to the researcher, an Automated Normalization program was written for Microsoft Excel using Visual Basic.

BACKGROUND

We shall discuss what is known and what is yet to be discovered in genetics research so that we may understand the role that cDNA microarrays has in science today. The Human Genome Project, of which the first draft has been completed [1], is a great step towards deciphering the blueprint for human life. We are close to having the full DNA sequence for the human genetic code. The central dogma of genetics research is that DNA is transcribed into RNA, and RNA is translated into proteins. The proteins have a function,

shape, and chemical composition as determined by their amino acid sequence, which is dictated by the DNA sequence of the gene. Each cell in the body only reads select portions of the entire genome at a time. The portions that are transcribed and translated into proteins determine many aspects of the form and function of a particular tissue type. The expressed DNA sequences or genes, in conjunction with the environment, ultimately determines what occurs within the cell.

However, we don't yet know the purpose and meaning of all segments of the genome. We are now ready to take the next step in understanding how the genetic code is interpreted into a particular function: Functional Genomics. Functional Genomics determines what genes in the genome are expressed in the various cells and tissues of the body, as well as the function of the protein products. Thus we hope to associate gene activity with the various cell and tissue types throughout the body, to determine the normal function of all genomic sequences, and relate their abnormal function to the cause of genetic diseases. Ultimately, we hope to be able to read and interpret the genetic code in terms of the function of the gene products, and with this information create a new level of understanding of the human organism.

INTRODUCTION TO MICROARRAY ANALYSIS

There is a tool to help accomplish the task of assaying gene activity in tissues: cDNA microarrays.[2][3] To create microarrays, cloned libraries of cDNA sequences are bonded to glass slides in a grid of spots micrometers in diameter, where each spot is a different sequence. Most of these cDNA sequences are uniquely identified in the GenBank database by accession number (see [Appendix A: Data Dictionary](#)). mRNA is extracted from the cell or tissue of interest and reverse transcribed into cDNA sequences with a fluorescent label. The slide is heated to denature the spotted cDNA pairs into separate strands. The labeled cDNA, often referred to as the probe, is also denatured and then hybridized on the glass slide to the array of spotted sequences. Finally, the microarray slide is scanned with lasers that cause the labeled cDNA to fluoresce, and an image of the slide is obtained using a CCD confocal microscope and read into a computer. This is the method for making microarrays currently used by the department of Pediatrics microarray facility at Oregon Health & Science University. There are other materials and technologies applied at other facilities, such as radioactive labeling of probes and other proprietary technologies. Our method is not the only method for creating microarrays, as others may use oligonucleotide sequences 25 base pairs long instead of cDNA or use radioactive labels instead of fluorescent labels. However, our method is the one that will be addressed in my thesis.

The rationale for this procedure is the following. The extracted mRNA is the direct product of the current gene expression activity in the cell line or tissue sample of interest. A one-to-one copy of the mRNA is made when it is reverse transcribed to cDNA. Thus,

the number of cDNA copies is proportional to the number of mRNA molecules in that particular cell. The cDNA copies will selectively bind to particular accession sequences from the clone library, those that make a complementary match in the target cDNA sequence. It is reasonable to assume that this match is the gene, or part of the gene, that originally coded for the mRNA found in the experimental cell line.[4] The amount of labeled cDNA hybridized to each spot can be measured by the intensity of the spot when the array is scanned into an image under fluorescence.

The challenge is how to make sense of all the information we gather with cDNA microarrays, and how to insure the reproducibility of those results. There is a large amount of quantitative data generated by cDNA microarray analysis, as each slide used in an experiment can generate 37,000 or more raw data points (counted from data stored in our microarray data warehouse). The 23 slide experiment series used to compare data processing and normalization methods in this paper contained 682,058 data points of raw data. Each spot printed on a microarray corresponds to a particular GenBank accession or clone library sequence from the genome.

First, let us consider what information we can obtain from a single spot, just a few micrometers in diameter on a microarray. When the microarray is scanned by laser fluorescence, a digital image is generated on the computer the scanner is connected to. The computer represents this image using a two dimensional arrangement of pixels of varying intensity. Each spot on the microarray, and the background surrounding it, is thus represented by a number of pixels. We use a 16-bit grayscale TIFF (Tagged Image File Format) image format to store this information. 16-bit means that we can store the intensity of each pixel as a number with a range of 2^{16} , that is a number from zero to 65535 ($2^{16} - 1$). In grayscale, this number translates to a shade of gray from black (zero) to white (65535). As a result, not only do we have a visual representation of the microarray on the screen as an image in shades of black and white, but this image translates into a quantitative method of extracting information on the fluorescence intensity detected by the scanner.

Since the microarray spot in question is made up of many of these pixels in the image, we can take an average of all pixels in the spot to get the mean signal intensity of the spot. We can make a similar calculation for the background region outside of the spot in order to correct the spot signal for background noise. For our purposes we use the median of these background pixels to minimize the influence of outlier pixels. These values, along with about 20 others used to record position and other statistical and quality information about the spot, are computed for us using proprietary software (AutoGene™ software by BioDiscovery, Inc. -- <http://www.biodiscovery.com/>). One commonly accepted method that we are currently using for correcting the spot signal for background noise is to subtract the median background from the mean signal for the microarray spot. This is done because after hybridizing the probe with the microarray, washing typically does not remove all of the fluorescent probe material from the portions of the slide where no cDNA was printed. The region of the slide where no cDNA was printed is referred to as the back-

ground. The assumption is that there is as much excess probe on each target spot in the array as there is on the background around the spot. The background can be uneven across the slide, as the probe can be washed unevenly from the slide surface. Likewise, the probe can be unevenly distributed on the slide during hybridization, influencing both the target spot and background intensity together. Therefore, the background is calculated within a given distance of each target spot, and the correction is applied to each spot individually.

All this is for just one spot on the microarray, corresponding to one genome accession from a clone library hybridized with our labeled probe. There are many of these spots in a microarray. Many of the microarrays generated by our lab have over 13,000 spots on a single slide. We extract all the information generated by our image analysis software for each microarray experiment and store it in an Oracle data warehouse (<http://www.oracle.com/>), along with their identifying accession numbers and grid placement information. The Oracle data warehouse and interface was implemented by my colleague, Bryan Olmstead, as the subject of his thesis project. Each microarray slide can be hybridized with two differently labeled cDNA probes that fluoresce under different color lasers. This allows us to compare the amount of hybridization of two different cDNA probes on the same slide, such as a control probe versus an experimental one.

RESEARCH QUESTION

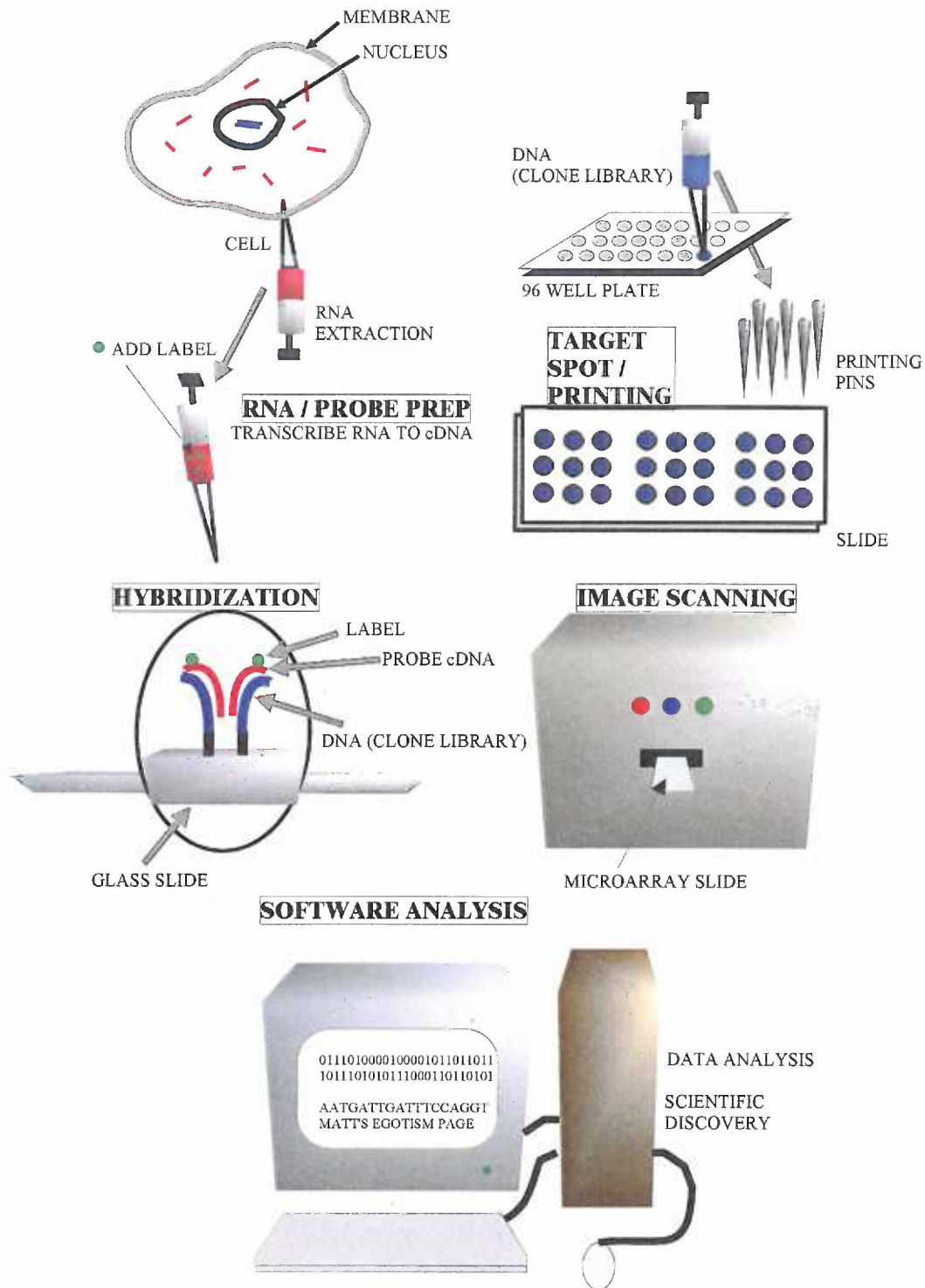
This thesis addresses the following research question: Can we generate a consistent and meaningful report of microarray gene expression results, correcting for systematic sources of variation, and ensure that the results are valid and reproducible for the genetics researcher? Our answer to this question is yes, we could and we have produced a microarray reporting tool that has been useful to the researcher.

Most genetics researchers do not want to be directly involved with all of the numerical data generated by microarray analysis, as the amount of information generated is overwhelming. The researcher is mainly interested in what genes are up-regulated or down-regulated in expression when compared to the same gene with a different treatment or in other cells and tissues. Researcher can use microarrays to answer questions about what genes share the same (or opposite) patterns of expression in different treatments or tissue types, and are therefore possibly linked in function. Genes of unknown function can be identified by comparing their expression level with a treatment response or with a related gene whose function is known.

In my role as a bioinformatics technician, I am concerned with how I can provide a useful interpretation of microarray data for the genetics researcher. More importantly, can we control or correct for variations in the data sufficiently to ensure the data is reproducible, or at least identify when and how problems occur? The researcher needs to know that the results they generate from microarray experiments will be the same if they or

anyone else were to do the experiment over again. They need to be able to identify and reduce or eliminate sources of error in their data when and where they occur as early in the process as possible. We are working to produce automated methods of analyzing raw data from microarray experiments, and generating a standard format for reporting practical results to the genetics researcher for future scientific discovery.

FIGURE 1: MICROARRAY PROCESS



SOURCES OF VARIATION BY PHASE

A microarray experiment is a complex process, and as such there are many possible sources of variation in the data produced. A diagram of these potential sources of variation, both from our work and others [5], is presented in [Figure 1](#) and summarized in [Appendix A: Data Dictionary](#). This diagram divides the potential sources of variation into groups relating to the different steps in the process of producing and interpreting microarrays. There are five phases in the process of generating microarray results: mRNA and probe preparation, printing of cDNA target spots on the slide, hybridization of the probe to the target spots, image scanning of the fluorescent labeled microarray, and software analysis of the image generated.

RNA AND PROBE PREPARATION

The probe preparation is subject to the following factors of variability: mRNA is very sensitive to degradation, reverse transcription can produce cDNA of varying length, and the different fluorescent labels used for two channel analysis (one label for each of two probes) can and do differ in sensitivity to the lasers used to excite them.

TARGET SPOT AND PRINT QUALITY

The target spot and print quality of the microarray can vary as follows: PCR amplification of the clone cDNA is difficult to quantify; the cDNA is dissolved in such small volumes of solution that it can evaporate while printing if it is not properly humidified; the pins used in printing can wear out or otherwise cause different quantities of cDNA to be printed; and irregularities in the size and shape of the printed spots can affect the spot detection later.

HYBRIDIZATION

Hybridization introduces a number of factors for variation: the efficiency of hybridization is influenced by temperature, time, buffering, adenine-thymine vs. cytosine-guanine content, and overall probe quality; there can be non-uniform distribution or non-uniform hybridization performance of probe across the slide surface; non-specific hybridization can occur between the probe and a target spot; and there is the possibility that the target spot could be saturated by the probe, eventually resulting in the underestimation of the mean signal for the spot.

IMAGE SCANNING

Different settings for PMT and laser intensity when scanning the slide into an image can cause spot intensity to vary. When those settings are too high, signal saturation can occur.

SOFTWARE ANALYSIS

Finally, the software analysis may fail to properly identify and quantify valid spots on the microarray with the available spot finding methods. The algorithm, parameters, and grid layout are all potential sources of error.

SUMMARY

There are five phases in the microarray process: RNA and probe preparation, target spot printing, hybridization, image scanning, and software analysis. The probe is prepared by extracting mRNA from the cell, reverse transcription of the mRNA to probe cDNA, and fluorescent labeling of the probe. The target cDNAs from the clone library are printed as spots on a glass slide. The probe cDNA (reverse transcribed from the original cell mRNA) is hybridized with the printed target spot cDNA on the slide. A CCD confocal microscope attached to a computer scans the slide with lasers that cause the fluorescent label to glow. Finally, the scanned image of the microarray slide is analyzed with various software to determine and report gene expression results back to the researcher. Each step in the process has various potential sources of error which should be accompanied with quality assurance and quality control measures to ensure the validity and reproducibility of the resulting microarray data.

INTRODUCTION TO DATA PROCESSING AND NORMALIZATION STRATEGIES

A combination of established procedures and proper normalization and data preparation methods during software analysis can help to minimize error inherent in the microarray process. The DNA microarray is a recent development and a powerful tool for genetics research. Microarrays are used for gene expression analysis, determining which genes are being actively transcribed into mRNA within the cell, as well as quantifying how much transcription is taking place. A microarray experiment is a complex process, and as such there are many possible sources of variation in the data produced [5] as discussed in the previous section. Here we shall discuss the various methods of data preparation and normalization used in the software analysis phase of the microarray process.

**YANG, Y.H. ET AL. NORMALIZATION FOR CDNA MICROARRAY
DATA, SPIE BIOS[6]**

This article produced from the combined efforts of the Department of Statistics and Department of Molecular and Cell Biology from the University of California at Berkley, the Department of Biochemistry from Stanford University, and the Division of Genetics and Bioinformatics from the Walter and Eliza Hall Institute in Australia [6] describes several common approaches to the normalization of microarray data. Three common approaches to normalization methods used in different types of microarray experiments are discussed: within-slide, paired-slides for dye-swap experiments, and multiple slide normalization. The paper also evaluates three different sets of genes commonly selected for normalization purposes: all genes on the array, constantly expressed or "house-keeping" genes, and spiked control spots or titration series on the slide (not necessarily genes, could be control sequences or blanks). This group evaluated the normalization methods using what they call an M vs. A plot. The M axis on this plot is the log intensity ratio $M = \log_2(R/G)$ where R and G are the red and green dye channels of a two dye experiment respectively. The A axis of the M vs. A plot is the mean log-intensity $A = \log_2 \sqrt{RG}$. This group claims that this method of comparison gives a more realistic sense of concordance than simply plotting the log-intensity of one dye vs. the other.

Yang *et al.* found that each method they investigated had advantages and disadvantages, and their utility in the analysis of microarray data was dependant on experimental design. They focused on methods that normalized data based on position, or print-tip-group. However this was most useful in correcting for differences in two-channel experiments, where two probes with different color fluorescent labels (usually red and green) are hybridized on the same microarray slide at the same time. They also noted two assumptions that their methods are based on: only a few genes in the experiment should show a significant change in expression between samples, and the expression levels of up and down regulated genes should have symmetry. Our slidewise normalization method shares the first of these assumptions, however we have not yet investigated the impact of the second on our methods.

**SCHUCHHARDT, J. ET AL. NORMALIZATION STRATEGIES FOR
CDNA MICROARRAYS, NUCLEIC ACIDS RESEARCH [5]**

The Institute for Theoretical Biology and Max Planck Institute of Molecular Genetics of Berlin, Germany[5] evaluated multiple strategies to control and correct for "systematic and stochastic fluctuations" in microarray data. This group identified and listed sources of noise in spotting and hybridization, many of which are included in the sources of variation listing described in the introduction of this paper. The observation was made that duplicate spots printed on the same slide showed significantly better correlation ($C=0.90$) than

duplicates printed on separate slides ($C=0.76$). The investigators concluded that variations for same slide comparisons was due to random fluctuations in target volume. Variations for comparisons across slides was attributed to differences in hybridization efficiency, unequal distribution of probe, and image processing factors.

The microarray for this study was printed using a 384 pin gridding head. This method produced 384 meta-grids with 36 spots in each grid (6x6). The radioactively labeled probe was obtained from reverse transcribed mouse tissues, and then a dilution control series was printed using *Arabidopsis thaliana* clones. Each meta-grid included four blank (empty background) spots and two spots from various mouse clones.

Four different normalization methods were tested. These methods are listed in order of increasingly accurate classification of signal intensities compared to differentially diluted control clones:

1. No normalization
2. Slide-wise normalization: divide each target spot by the average intensity of all control spots on the slide.
3. Pin-wise normalization: divide each target spot by the two constant control spots in the same meta-grid.
4. Average pin-wise normalization: slide-wise normalization of target and control signals then dividing the average target by the average control.

The performance of these methods were compared using the percent match of replicate known dilution levels for printed spots in training and test groups. Schuchhardt et al. found that average pin-wise normalization performed better than all others, pin-wise was second, followed by slide-wise, and no normalization was last.

**NEWTON, M.A. ET AL. ON DIFFERENTIAL VARIABILITY OF
EXPRESSION RATIOS: IMPROVING STATISTICAL INFERENCE
ABOUT GENE EXPRESSION CHANGES FROM MICROARRAY DATA,
JOURNAL OF COMPUTATIONAL BIOLOGY [7]**

The University of Wisconsin uses an *Escherichia coli* cDNA microarray to demonstrate a method of data processing for the comparison of relative expression using empirical Bayesian analysis. This article addresses the fact that calculating relative expression using standard fold change ratios may have a different interpretation for low signal levels than they do for high signal levels. Other groups have found evidence of greater relative variability in data at lower expression levels when compared to higher levels of expression [8]. This is because ratio data does not address the effects of range, where low values that are essentially so close to zero as to be considered blank will show more than the standard 3-fold change in expression when a ratio is calculated. For example, both $0.009 / 0.003 = 3.00$ and $900 / 300 = 3.00$ result in a 3-fold change in expression. However, the difference

of $0.009 - 0.003 = 0.006$ is essentially zero, while the difference of $900 - 300 = 600$ is more than zero. As a result, the authors of this article suggest that raw intensity ratios may be prone to error. In our lab, we have observed this effect consistently when graphing and comparing duplicates, where every graph shows more scatter at low values (see Figures 2 and 3). Ratio comparisons between treatment and control data sets, particularly when performed using two-channel analysis, do not address this issue.

The authors of this article suggest a solution to this problem, detailing a statistical method of processing cDNA microarray data prior to normalization using empirical Bayes estimates. Rather than relating expression data in terms of fold changes and simple ratios, empirical Bayesian analysis basically generates a set of hypothesis tests, one for each gene. The null hypothesis is that there was no change in expression (the measured intensity of control and treatment spots is equal), and the alternative hypothesis is that there was a measurable change in expression (the intensities are unequal). Thus, any comparison that falls outside of a suitable confidence interval has a significant change in expression for that gene.

RICHMOND, C.S. ET AL. *GENOME-WIDE EXPRESSION PROFILING IN ESCHERICHIA COLI K-12*, NUCLEIC ACIDS RESEARCH [9]

This article was an earlier work from the University of Wisconsin using *Escherichia coli* cDNA microarrays. The paper compares the results from radioactively labeled nylon membrane arrays to fluorescently labeled glass microarrays. In this case, the normalized data was calculated as a percentage of total target signal after background subtraction. Each method had similar expression results, but the fluorescence microarray data was more reproducible (had less variability among duplicates).

DELENSTARR, G. ET AL. *ESTIMATION OF CONFIDENCE LIMITS OF OLIGONUCLEOTIDE ARRAY-BASED MEASUREMENTS OF DIFFERENTIAL EXPRESSION*, PROCEEDINGS OF SPIE [10]

Agilent Technologies provides a comparison of data processing and normalization methods on human oligonucleotide microarrays. They developed two methods of background correction:

1. Local nearest neighbor: Target signal minus average background of 3x3 grid surrounding target spot.
2. Negative control features: Target signal minus average signal of replicate negative control features using probes determined to have minimal hybridization to target spots.

When the duplicate graphs of these two methods were compared, the investigators found that the negative control features showed less variability than the local nearest neighbor method.

Because Agilent Technologies works with 25 base oligonucleotide microarrays, cross-hybridization becomes a significant source of error. They address this issue with the development of deletion control probes. Microarrays using 25 base oligonucleotides have base 12 removed from the probe sequence, forming a 24 base deletion control. This is subtracted from the "perfect complement" probe, which is sensitive and specific to the target sequence, resulting in what they describe as a "net perfect match signal". The probe and target sequences used in cDNA microarrays are thousands of bases long, such that cross-hybridization is not a significant source of error and can be ignored.

To compare relative expression, Agilent Technologies calculates the ratio of net perfect match green and red signals, forcing the average log ratio to zero. Error propagation methods are used to determine the log ratio error and converted into a p-value. The log value significance can be evaluated based on a chosen threshold p-value.

SUMMARY

The majority of laboratories using microarrays have had to develop and test their own methods for data processing and normalization of microarray data with varying degrees of success, yet no one has found a definitive answer to the question of what methods work best. Schuchhardt et al. found that replicates printed on the same slide matched each other more consistently than they did when printed on different slides, suggesting a significant amount of variability inherent in the process of printing multiple microarray slides, which makes comparisons between slides a challenge. They also found that of the four normalization methods they tested, average pin-wise normalization performed the best on replicate spots in a dilution series. Newton et al. suggests that empirical Bayes estimates are more accurate predictors of the gene expression from microarray data than simple fold-change ratios. Richmond et al. used percentage of total signal as their measure of gene expression and found that fluorescence microarray data was more reproducible than radioactively labeled nylon microarrays. Delenstarr et al. report good results with their negative control features for background correction. They worked around cross-hybridization issues inherent in oligonucleotide microarrays, and compared relative expression using log ratios forced to zero. Delenstarr et al. also worked with statistical probability measures rather than simple fold-change ratios.

METHODS

QUALITY ASSURANCE

To provide quality assurance for cDNA microarray processing in our lab, a series of protocols have been generated to describe the laboratory process. The protocols are published in both Microsoft Word 97 and Adobe PDF (Portable Document Format) formats and are available on our GeneViews Web page at <http://medir.ohsu.edu/~gene-view/pages/protocols.html> . Updates to these protocols are made on the Web for current reference.

DATA PROCESSING AND NORMALIZATION

In order for cDNA microarray data to be useful to the researcher, some data processing and normalization has to be applied to the raw data extracted from image analysis to meaningfully represent gene expression information. Normalization refers to a correction or weighing factor applied to reduce systematic variability from a given data set. Data processing refers to all other forms of filtering or transformation of data.

The raw microarray data consists of a median signal, mean background, selected flag, and position or meta-grid location for each accession on the array. The local corrected value is computed for each accession as the mean signal minus the median background. The local corrected value is then normalized.

Several normalization methods were evaluated and compared in the course of writing an automated analysis and reporting tool for researchers performing cDNA microarray experiments in our lab. The normalization method should correct raw data for systematic variations in the microarray process, as mentioned in the Background section of this paper, in order to improve the validity and reproducibility of the results. The normalization methods evaluated were as follows:

1. Standards or "housekeeping" genes

For gene i in a given microarray, let x be the original local corrected value,
 Y be a set of "housekeeping" genes where $Y \subset X$,
and n be the normalized signal intensity.

Then for each gene (spot) in the microarray,

$$\frac{x_i}{\overline{Y}} = n_i$$

2. 20% trimmed mean slidewise

For gene i in a given microarray, let x be the original local corrected value, and n be the normalized signal intensity.

Then for each gene (spot) in the microarray,

$$\frac{x_i}{\overline{X}_{trim}} = n_i, \text{ where } \overline{X}_{trim} \text{ is the 20\% trimmed mean.}$$

3. Gridwise (Mean)

For gene i in a given microarray, let x be the original local corrected value, Z be the set of genes printed in the same meta-grid as x where $Z \subset X$, and n be the normalized signal intensity.

Then for each gene (spot) in the microarray,

$$\frac{x_i}{\overline{Z}} = n_i \text{ or } \frac{x_i}{\overline{Z}_{trim}} = n_i, \text{ where } \overline{Z}_{trim} \text{ is the 20\% trimmed mean.}$$

4. Log ratio slidewise

For gene i in a given microarray, let x be the original local corrected value, and n be the normalized signal intensity.

Then for each gene (spot) in the microarray,

$$\log x_i - \log \overline{X} = n_i$$

Normalization using standards, or what are commonly called "housekeeping" genes, is based on the assumption that there is a set of accessions printed on the microarray that should show no change in expression between experimental treatments. These accessions are usually regulatory genes for the cell line. Each local corrected spot is divided by the mean, median, mode, or trimmed mean of all standards on the slide or array. One additional method is to filter based on "good" standards, those standards that pass a duplicate test, before averaging those standards for normalization. For the purpose of this thesis, I compared both a trimmed mean of standards and an average of "good" standards for the same experiment to a trimmed mean slidewise normalization.

The second normalization we investigated was the 20% trimmed mean slidewise method. This method assumes that most of the accessions on the microarray slide will not show a significant change in expression between treatments. The top 10% and bottom 10% of all local corrected values on the array are removed before computing the mean to reduce the influence of outliers. Each local corrected value is then divided by the 20% trimmed mean of all accessions on the slide. We tried a couple of other trimmed means (10 and 30 percent) before choosing 20%. In this way, enough outlier spots were excluded so as not to bias the mean and still keep enough genes in the calculation for the mean to be descriptive of the given microarray. This method was chosen as the default method of normalization in our lab, and has been compared to each of the other methods in this paper.

Our third method of normalization was the gridwise method using the mean of accessions grouped by meta-grid. The local corrected value for each accession was divided by the average of local corrected values in each meta-grid. This method assumes that the average level of expression in each meta-grid should not be significantly different from another.

The final normalization method we evaluated was a slidewise log ratio. A log base 10 was applied to each local corrected value. The mean of the log values was computed for the entire slide, and this was subtracted from the log of the local corrected value for each accession to obtain the log ratio. The log ratio could then be converted back to arithmetic scale (by computing the 10th power of the log ratio) for use in determining fold change, as discussed later in this paper. This method is similar to computing a geometric mean for each slide.

After normalization, we can either continue with our standard data processing methods, or export the normalized data to an external statistics or microarray analysis program, such as ArrayStat™ by Imaging Research, Inc. ArrayStat™ can be used to determine gene expression using statistical significance by performing an independent z-test for each array, corrected using a step-down Bonferroni. If the normalized value for an accession falls outside of the computed confidence interval, it is flagged as showing a statistically significant change in expression.

If we continue with our data processing, all the normalized values are floored to a minimum value to reduce the variability inherent with low levels of expression. Any normalized values which are less than the floor value can be considered as having no significant level of expression, which we then set to be equal to the floor value for further analysis. We typically set a default floor value for normalized data to 0.10. When using the 20% trimmed mean method of normalization, this is equivalent to 10% of the trimmed mean for all spots on the slide.

A duplicates comparison is performed if replicate slides were printed as a part of the microarray experiment (highly recommended). Duplicates are tested to determine if they fall within a given minimum fold change, usually set to 3.00 fold. Fold change is used as an estimate of differential expression between samples. If the duplicates for an accession are outside of the given fold change (greater than 3.00 fold different), the pair are flagged as bad for filtering and quality control. Then a constant fold change, usually set to 2.00 fold, is applied. In order to pass the constant fold change data filter (not be flagged as bad and excluded from the results), the accession must show a change in expression that is equal to or more than the constant fold change value between all experimental conditions (arrays or slides) in the series. This filter is based on the assumption that the researcher is only interested in those genes that show a significant change in expression during the course of the microarray experiment. In our lab, the constant fold change has been specifically requested as an option for filtering and analysis.

After any duplicates have been processed and averaged together, ratios are calculated for each treatment to control or treatment to treatment comparison the researcher needs from the experiment. These ratios are interchangeable with fold change as defined in the attached Data Dictionary (Appendix A) for describing the relative level of expression for each accession. A list of accessions is generated, typically filtered with a minimum of 3.00 fold change in expression for each accession, with gene annotations such as the gene name for the accession defined in the UniGene database.

QUALITY CONTROL

I have created four simple measures of quality that can be reported to the researcher to help determine the usefulness of their reported expression data:

1. 20% Slidewise trimmed mean (used for normalization)
2. % Saturation (above ceiling)
3. % below Floor (reported no expression)
4. Duplicate tests (after normalization)

The 20% slidewise trimmed mean is a measure of the average intensity of all target spots on the slide. Percent saturation is an indicator of how much data loss occurs due to measurement limitations at the high end of the range (2^{16} or 65536). This is calculated as total number of accessions whose local corrected values are above a given ceiling (i.e. 55k) divided by the total number of accessions on the slide, multiplied by 100. The percent below floor is the reverse of the percent saturation, indicating how much of the data does not show any measurable level of expression. This is calculated as the number of accessions below a floor value divided by the total number of accessions on the slide, multiplied by 100. A related quality control measure could be used in the near future to evaluate the overall intensity of blanks, control spots where no cDNA was printed on the array.

A low overall intensity rating and high percent below floor can indicate failed hybridization on the microarray, or an underdeveloped image obtained from the laser scanner. A high overall intensity rating and high percent saturation can indicate excessive hybridization on the microarray or an overdeveloped image obtained from the laser scanner. If all three measures are high, this can indicate the range of spot intensity on the microarray exceeded the limitations of the measurement device (laser scanner).

The final quality measure, duplicate tests, can be displayed as a scatterplot of one vs other, the total number of bad duplicates on a slide, and the percentage of bad duplicates on each slide. These measures give an indication of the reproducibility of the normalized data. If there is a high number or percentage of bad duplicates, or the majority of data points on the duplicate graph do not fall between a slope of 3 and 1/3 on arithmetic scale

(or come close to a slope of one on a log scale), then there is likely a problem with one or both of the duplicate slides.

REPORTING AND EXPERIMENTAL DESIGN

We wrote an Automated Normalization program to automate established cDNA microarray data processing and normalization procedures and generate results useful to researchers without the direct assistance of a bioinformatics technician or statistician during the initial analysis. Since the researchers in this lab used Excel primarily to analyze their experiment results, I wrote the program using Visual Basic as the built-in programming language for Microsoft Excel. The program started as simply a collection of code and fix macros to automate some of the repetitive steps I performed in doing the analysis myself. But as the macros and scripts became more advanced, I decided to combine them into a single program that typical researchers could understand and operate on their own to produce the same results. Here is a list of the features offered by the Automated Normalization program in its current version:

- Online documentation updated regularly with help buttons at every step of the process linked directly to the GeneViews Web page.
- http://medir.ohsu.edu/~geneview/pages/protocols/AutoNorm_Protocol.html
- Import raw microarray data generated by AutoGene™ by BioDiscovery from core facility BARRY Data Warehouse export or AutoGene™ text file export.
- Calculate signals locally corrected for background.
- Perform users choice of accepted normalization schemes on locally corrected data. Currently 20% trimmean slidewise normalization scheme is supported, others to be added as developed and tested. Meta-grid normalization is available, but unsupported.
- Compare duplicate microarray slides, flagging target spots whose duplicate does not match (shows a change in expression for same sample and treatment). Average duplicates, allowing researcher to filter out flagged duplicates if desired.
- Compare all arrays / slides in microarray experiment, flagging spots with a constant fold change (does not show a significant change in expression across entire experiment).
- When available, add quality indicators / flags to reported microarray data.
- Calculate ratios (difference for log transform, or other methods of comparison) between experimental conditions to determine change of expression.
- Add GenBank or ResGen gene names to accession numbers used in microarray.
- Export results to GeneMaths (previously GenExplore) by Applied Maths for cluster analysis.

SUMMARY

We came up with four normalization methods to compare in our lab: mean of standards, 20% slidewise trimmed mean, mean gridwise, and slidewise log ratio. To automate the process of normalization and data processing as outlined in the methods section, an Automated Normalization program was written in Visual Basic to be run on Microsoft Excel spreadsheets. This program reports gene expression results in fold change ratios, and reports the quality of microarray slides with duplicate graphs and overall slide intensity measures. The 20% slidewise trimmed mean and mean gridwise normalization methods and all data processing steps were implemented in the Automated Normalization program. The log ratio method was tested by comparing results from the external application ArrayStat from Imaging Research, Inc. to the 20% trimmed mean results using our program.

RESULTS

QUALITY ASSURANCE

The laboratory protocols for the major steps of cDNA microarray preparation are published and maintained at <http://medir.ohsu.edu/~geneview/pages/protocol.html>. Appendix B contains a hardcopy of the most current versions of those protocols as of this printing.

COMPARISON OF NORMALIZATION METHODS

We compared four normalization methods in our lab: standards or "housekeeping" genes, 20% slidewise trimmed mean, mean gridwise, and slidewise log ratio. For the mean gridwise and "housekeeping" gene methods, we performed a simple comparison of duplicate graphs with the 20% slidewise trimmed mean method to assess the reproducibility of each. For those comparisons, our conclusions on validity were based on observation instead of more extensive comparison due to time constraints. For the 20% slidewise trimmed mean and slidewise log ratio method comparisons, we compared the results of using our Automated Normalization program for the trimmed mean vs. the ArrayStat program from Imaging Research, Inc. for the log ratio method. In order to compare log scale data with statistical tests for significance to arithmetic data with fold-ratio comparisons, we plotted graphs of the local corrected data with genes showing differential expression in one, both, and neither of the two methods. Then we looked at a subset of genes where the two methods disagreed to see if we could determine the reason for their differences by observation.

STANDARDS OR "HOUSEKEEPING" GENES

Variations of this method were compared with slidewise trimmed mean normalization methods. The two major variations we tested were the trimmed mean of all standards by slide and the trimmed mean of "good" standards only. The "good" standards were a subset of all standards that had a measurable and constant level of expression between slides. The problem was, the list of "good" standards, genes whose expression was not supposed to change between experiments, actually came out with significantly different values in each experiment. The list of "good" standards always changed, so we concluded that we did not have any true standards or "housekeeping" genes to measure, that is that the so called "good" standards changed expression between experiments and conditions. This violated the method's assumption that regulatory genes did not significantly change expression

between arrays or experimental conditions, so we rejected this as a useful normalization scheme in our lab. In addition, when we compared duplicates normalized with "good" standards to those normalized with the 20% slidewise trimmed mean, we found the trimmed mean method performed better, as shown in Figure 2.

FIGURE 2: "GOOD" STANDARDS DUPLICATES

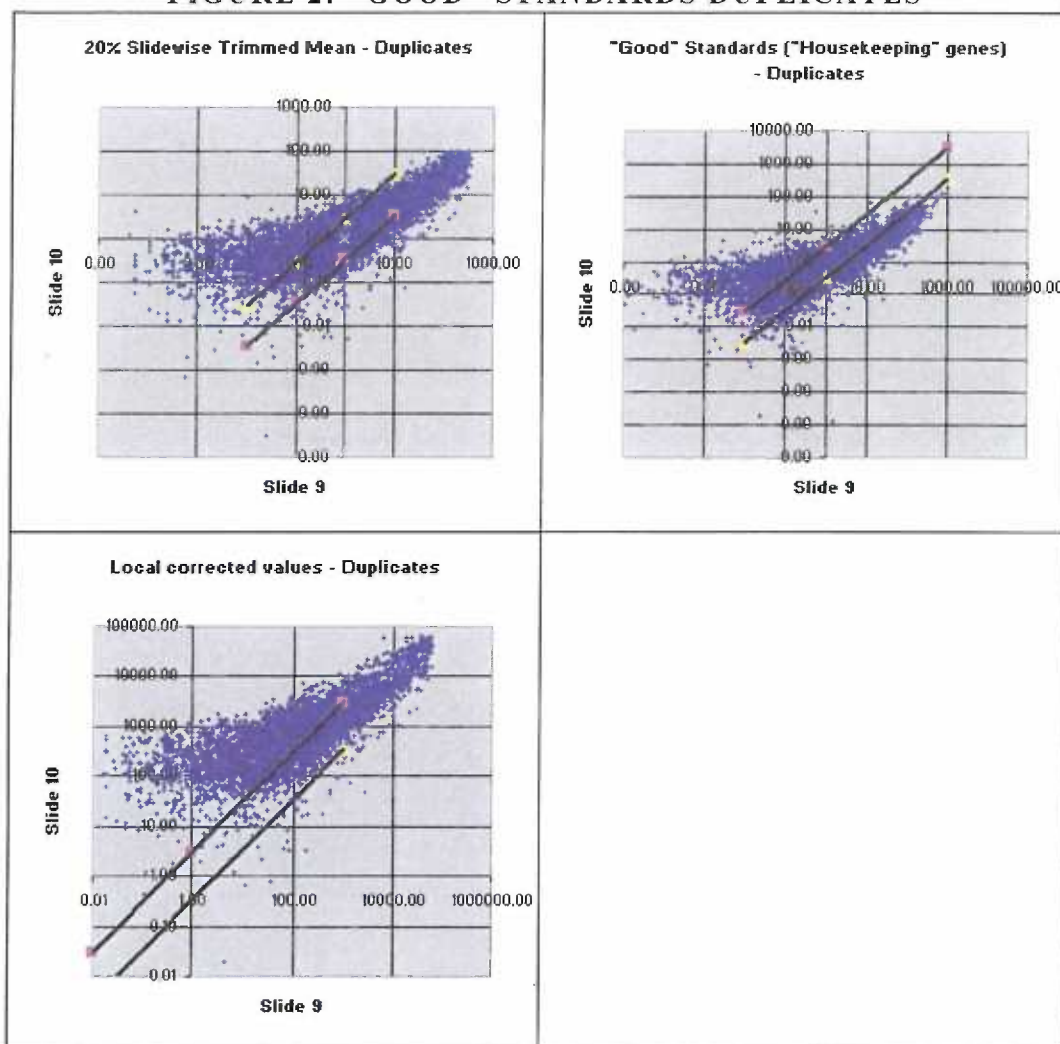


Figure 2: Above are duplicate comparison graphs of the "good" standards or "housekeeping" genes (upper-right), the 20% trimmed mean method (upper-left), and the local corrected values with no normalization. The line connecting the yellow triangles is the upper bound for 3-fold change, while the line connecting the purple squares shows the lower bound for 3-fold change in expression. All dark blue data points plotted between the 3-fold lines are considered to be good duplicates for publication purposes. A good normalization method will shift the data points to the center of the graph. For this slide, there was poor hybridization of the probe to the target, yet the 20% slidewise trimmed mean method centered the data in the linear region (successfully hybridizing expressors at upper end of graph) better than the "good" standards or "housekeeping" method as shown above.

20% TRIMMED MEAN (SLIDEWISE)

The 20% slidewise trimmed mean method has been the most successful and simplest normalization method that we have tested overall. In addition, this normalization can be taken with no additional data processing from within Excel and exported into the ArrayStat™ statistical microarray analysis application from Imaging Research, Inc. The normalized data can then have the change in gene expression measured for statistical significance using step-down Bonferroni and independent z-tests for each accession on the array. This is more accurate than simple fold change ratios as it includes factors such as the range and variance of independent data points in the analysis. However, fold change ratios do give a reasonably good estimate of expression levels with relatively simple computation.

The assumption that the 20% trimmed mean method relies on, that most target spots printed on the array do not show a significant change in expression between arrays, has not been disproved so far in the experiment sets we have seen in the lab. Duplicates printed on separate slides show generally good results when graphed together with this method. See Figure 3.

FIGURE 3: 20% SLIDEWISE TRIMMED MEAN DUPLICATES

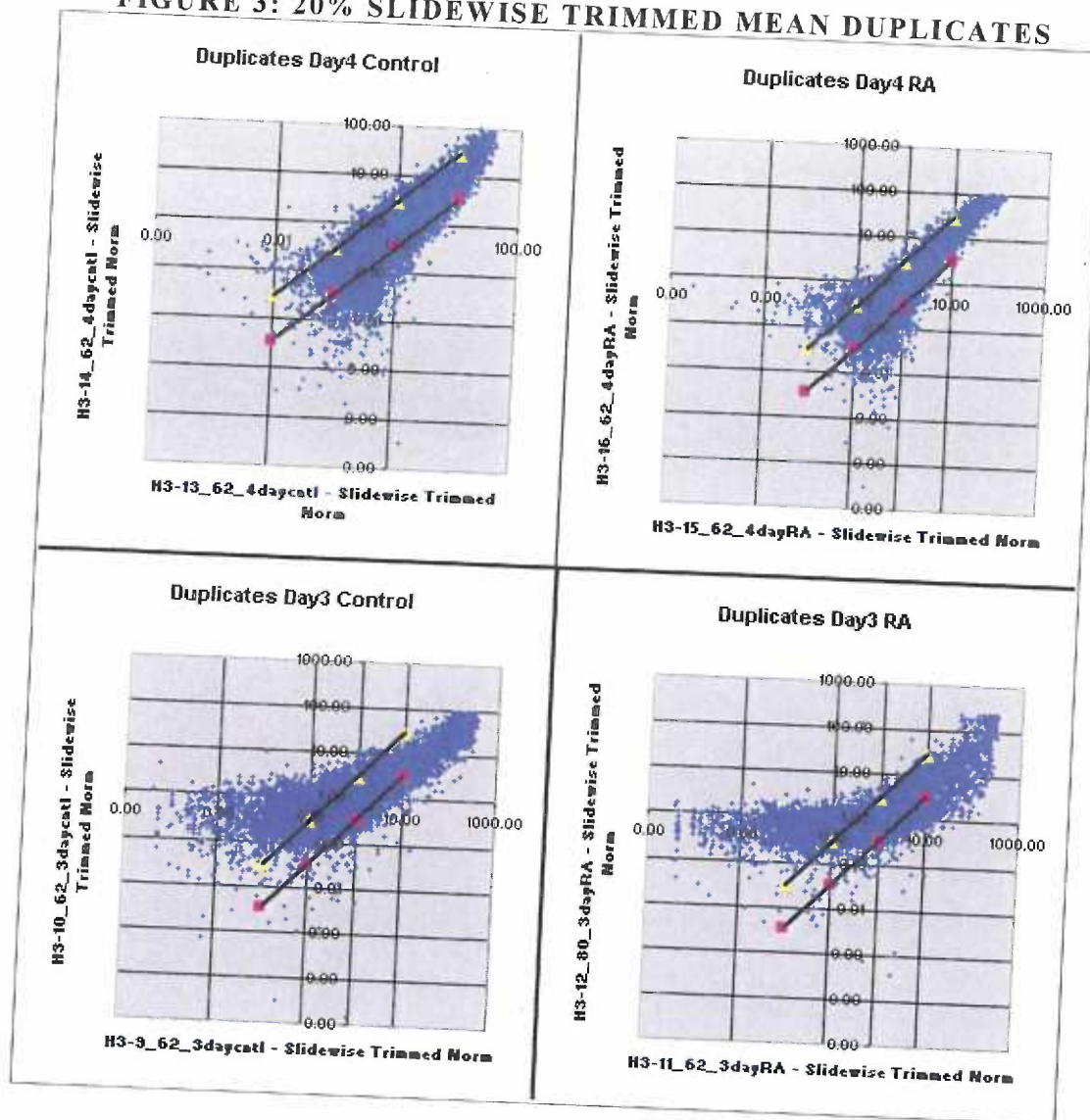


Figure 3: The control and RA treated duplicates for Day 4 (upper left and upper right) show normalized values graphed vs their duplicate on a logarithmic scale with good data. The Day 3 duplicates (lower left and right) are also duplicate graphs, but with lower quality data. Note that on the bottom two graphs, the low values fall above the linear region of no change in expression. We later found using the quality control measures we developed that this was likely the result of poor hybridization conditions as these slides had a high percentage of data points below the floor value and the research technician mentioned the slides were dark no matter how high the scanner settings were (gain and PMT).

GRIDWISE

The gridwise normalization method was rejected after analysis in our lab. This method was similar to the pin-wise normalization method used in the Schuchhardt *et al.* [5] paper.

Schuchhardt's group used a 384-pin printing head to print 384 meta-grids (one meta-grid per pin) with 36 spots in each, for a total of 13,824 target spots per slide. Each meta-grid of their experiment used a repeating pattern of blank, constant control, dilution control series, and two mouse clone spots. This arrangement should result in meta-grid groupings that show similar levels of expression, thus this method of normalization makes sense for the type of experiment used in the Schuchhardt *et al.* paper.

Most of our experiments, however, do not share this type of format. We use a 16-pin printing head and typically print 48 meta-grids (four meta-grids per pin) with 288 spots each, for a total of 13,824 target spots per slide. Also, we do not have a repeating spot arrangement in each grid, rather our control and treatment spots tend to be grouped into the same meta-grids. The gridwise method attempts to shift grids with different intensities together, instead of shifting different slide intensities together. We are not looking for changes in expression between grids, some of which may be mostly blank and others mostly expressed, rather we want to compare the change in expression between samples on different slides. Therefore, we chose to reject the grid-wise method of normalization as it does not appear to be useful for our typical experimental design. We still compared the duplicates of the 20% slidewise trimmed mean method to the mean gridwise method in Figure 4.

FIGURE 4: GRIDWISE VS. SLIDWISE

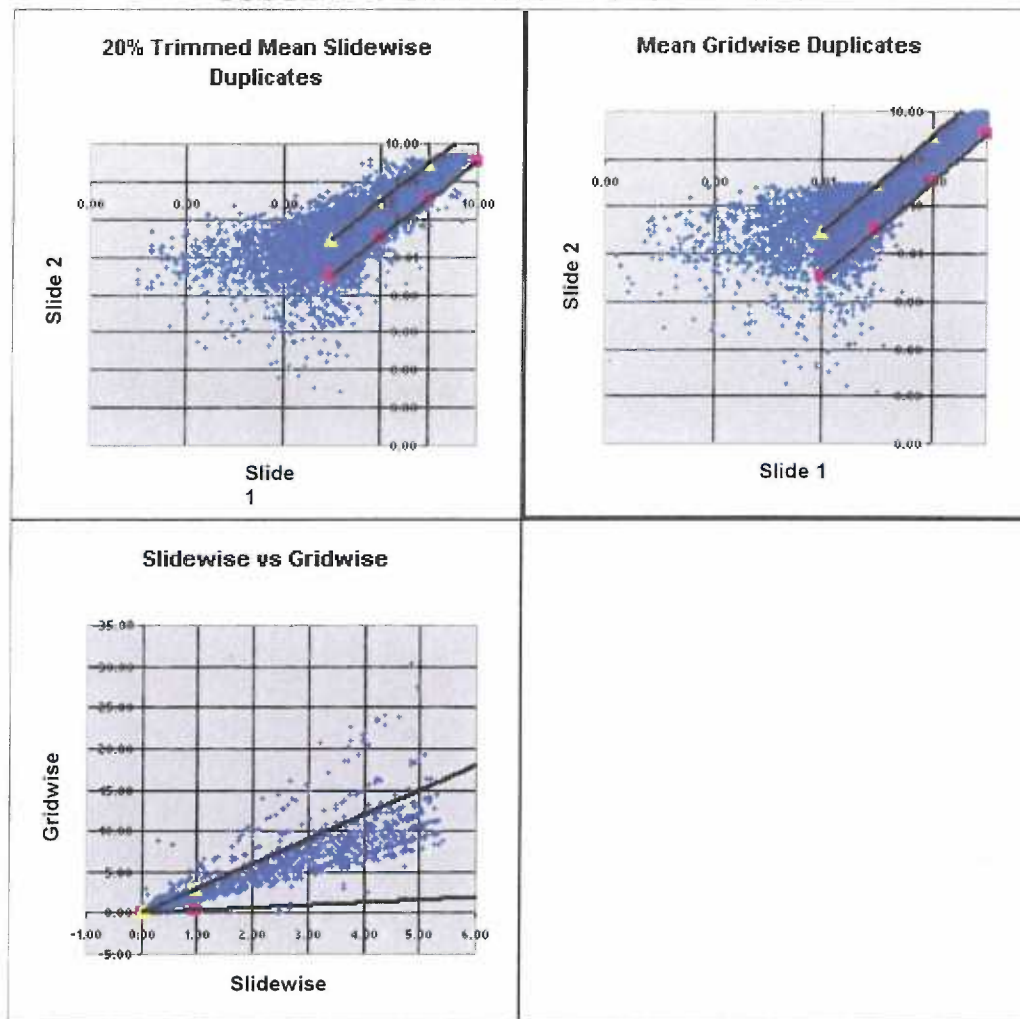


Figure 4: The duplicate graph of two replicate slides on a logarithmic scale using the 20% slidewise trimmed mean normalization (upper left) shows a more scattered distribution of data points when compared to the same two slides using mean gridwise normalization (upper right). When the same slide is graphed on an arithmetic scale with the slidewise method on one axis, and the gridwise method on the other (bottom left), we see that the data points group themselves to form multiple lines of different slopes. This is likely because the gridwise method attempts to shift grids with different intensities together, instead of shifting different slide intensities together. Since we want to make slides with varying hybridization efficiencies and laser scanning intensities more comparable, not the "blanks" printed in one grid with the "expressors" in another grid, the slidewise method is more valid than gridwise.

LOG RATIO

The log ratio required considerably more computation than the 20% trimmed mean method, but with little return. The results for log ratio normalization were somewhat comparable to the trimmed mean method, but the most notable difference was how the log

ratio tended to reverse the direction of fold change (up-regulation versus down-regulation of the genes) for small local corrected values. This is because the log base 10 of a value center on one instead of zero on the arithmetic scale. We found that for the log ratio to be useful for low intensity spots, it must either be adjusted to zero or a floor value must be set prior to performing the log transform. But since the 20% trimmed mean compared well to this method in all other respects, it was decided to drop further investigation into the log ratio normalization in Microsoft Excel. However, the results of exporting 20% slidewise trimmed mean data for statistical significance analysis on ArrayStat software from Imaging Research, Inc. showed some promising results. The ArrayStat software uses z-tests and additional steps for removing outliers to determine relative change in gene expression that may be more accurate than fold change ratios as discussed in the Newton *et al.* paper and the Introduction to Normalization Strategies section of this paper. However, this method is also more computationally intensive than standard fold change ratios. See Figures 5 and 6 for the slidewise log ratio vs. 20% slidewise trimmed mean comparison.

FIGURE 5: ARRAYSTAT VS FOLD CHANGE (EXCEL)

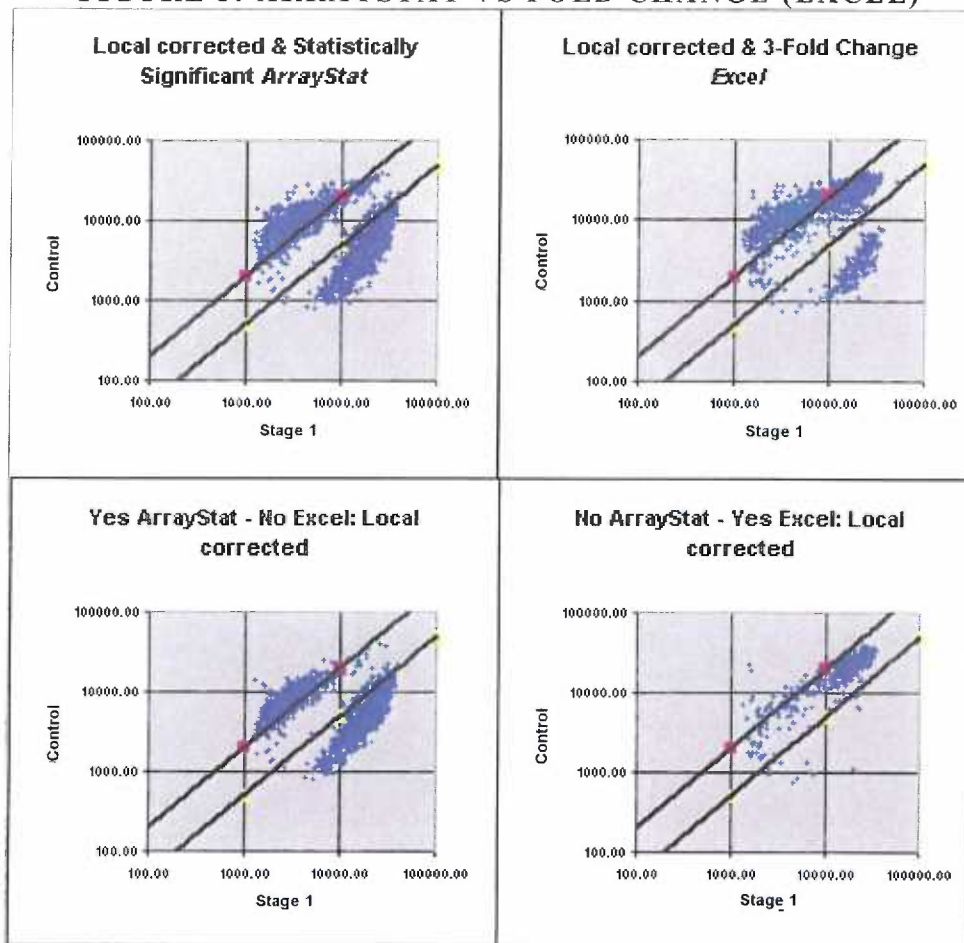


Figure 5: The data points in the graphs above were plotted using local corrected values, not normalized values to compare the results of two different normalization methods. A log ratio normalization was performed using the ArrayStat program, and a 20% trimmed mean normalization was performed using my Automated Normalization program in Microsoft Excel. In ArrayStat, we performed a log ratio normalization on the data before running a z-test to determine change in expression between pooled data for Stage 1 treated slides and the control slides. In Excel, we used the 20% trimmed mean normalization, averaged (pooled) the replicates, then determined the fold change ratio as average Stage 1 / average control, using 3-fold as the minimum significant change in expression. We chose to investigate to what extent the list of genes that ArrayStat determined were differentially expressed matched with the list of genes that our program in Excel found to be differentially expressed. Top left shows the local corrected value of genes showing a significant change in expression after conversion to log ratios using ArrayStat, while the top right shows the same results after slidewise trimmed mean was performed in Excel. Bottom left shows only local corrected spots that showed significant change in expression in ArrayStat, but not in Excel. Bottom right shows local corrected spots with significant change in expression in Excel, but not in ArrayStat. Thus, the bottom graphs show which genes each method disagreed on the decision for significant change in expression. The comparison continues in Figure 6.

FIGURE 6: ARRAYSTAT VS. FOLD CHANGE

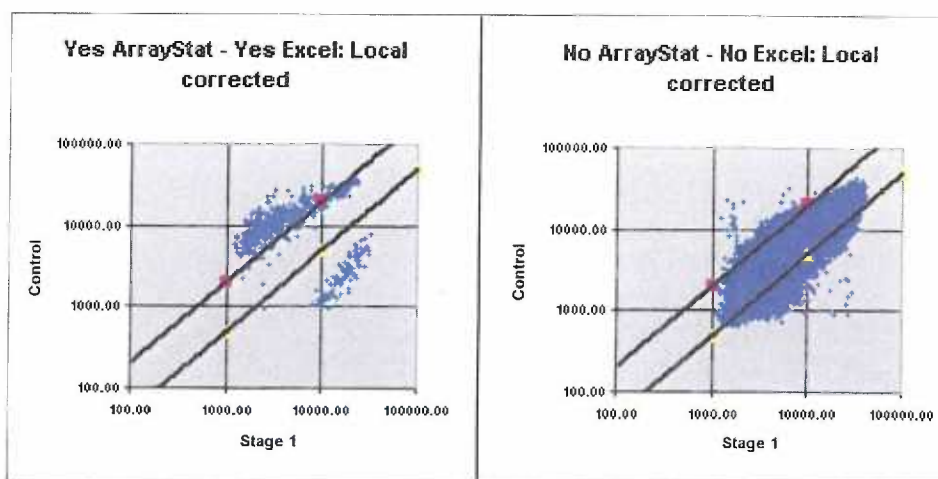


Figure 6: This is the continuation of the comparisons made in Figure 5. Left shows local corrected spots with a significant change in expression in both ArrayStat and Excel as described in Figure 5. Both methods were in agreement that these genes were indeed differentially expressed in the Stage I group when compared to the control. Right shows local corrected spots that both ArrayStat and Excel showed no significant change in expression. Those genes did not show differential expression in either data analysis method.

SUMMARY

We investigated four normalization methods: standards or "housekeeping" genes, 20% slidewise trimmed mean, mean gridwise, and log ratio. The standards and gridwise methods were only compared to the 20% slidewise trimmed mean method using duplicate graphs. However a complete side-by-side comparison of expression results was performed for the log ratio and trimmed mean methods. The 20% slidewise trimmed mean method is a simple and effective method of normalization for most of the microarray experiments we work with. The log ratio appears that with some correction, it could be comparable to the 20% slidewise trimmed mean method, but is more computationally intensive. However, it may be more accurate to use outside statistical applications, such as ArrayStat, to determine differential expression statistical significance tests instead of standard fold change ratios that may be error prone for low values of expression.

QUALITY CONTROL

There are four quality measures implemented in the Automated Normalization program.

1. Slide Intensity

a) 20% Slidewise Trimmed Mean

- b) Slidewise Mean
 - c) Slidewise Median
 - d) Slidewise Mode
2. % Above Saturation (55k)
 3. % Below Floor (500)
 4. % Bad Duplicates by slide

Additionally, the program can generate a series of duplicate graphs for visual comparison and verification of replicate data, such as those shown in Figure 2.

The first three quality measures are calculated using local corrected values before normalization, while the fourth measure and graphs are obtained from the worksheet generated during the duplicate comparison step of the program. The slide intensity measure provides a general measure of the distribution of spot intensities on the slide, using the mean, median, mode, and the 20% trimmed mean used in our lab for normalization. Since the range for spot intensity using 16-bit TIFF (Tagged Image File Format) images is from 0 to 65535 (2^{16} possible integer values), the local corrected value of 55,000 was selected to be the ceiling for the purpose of testing this part of the program. The ceiling value is defined as the point above which the spot is considered to be saturated, and a saturated spot will always underestimate the true value of expression, thus resulting in measurement error. As 55,000 is roughly the 85th percentile of the measurement range (0 to 65,535), and the 20% trimmed mean method removes values above the 90th percentile before averaging values across the slide, 55,000 appeared to be a reasonable ceiling value for evaluation purposes. The percentage of all spots on the slide that are above the given ceiling is reported in the second quality measure. Likewise, we chose the local corrected floor value of 500, where any spot falling below this floor value is considered to be the equivalent of a blank or nonhybridized, nonexpressing spot on the array. This value is about 0.8 percent of the measurement range (0 to 65,535), thus close enough to zero for the corresponding spot to be considered blank. The percentage of all spots on the slide that are below this floor value are given in the third quality measure. As both the ceiling and floor values are still under evaluation, we do not yet have a full assessment of the optimal ceiling and floor values for microarray experiments. Together, these three measures can give an estimate of how well developed each slide was during the imaging process or how well the hybridization performed across the slide.

The fourth measure is the percentage of duplicates in the entire slide that were flagged as "bad" within the fold-change entered by the user. This measure, combined with duplicate graphs, can be used as a measure of reliability across the experiment. The assumption is that duplicate accessions printed on different microarray slides under the same treatment should be of the same intensity or level of expression as its match, meaning the results are reproducible.

REPORTING AND EXPERIMENTAL DESIGN

The Automated Normalization program has been released into a working laboratory environment and tested by several research lab technicians performing microarray experiments to their general satisfaction. A link to my e-mail address has been provided on the title page of the online help for bug reports and suggested program changes. The program currently assumes that it is running on the laboratory intranet with access to shared network volumes available. It also assumes that raw microarray data is being imported from the BArry Microarray Data Warehouse (running on Oracle) created by coworker Brian Olmstead. For every step of the automated process, the program assumes that the currently open Excel worksheet is in the proper format for the step to be performed successfully. It is currently the user's responsibility to read the dialog boxes that describe the proper spreadsheet formats and format the sheets properly prior to running each step. Most steps can be successfully run in sequence without manually moving or editing the worksheet layout, but at this point the duplicate and ratio steps usually require some manual formatting of the spreadsheet before they are run.

Certain choices for experimental design will have an effect on how microarray data should be prepared and analyzed. Common choices for experiment designs include control / treatment experiments, time course experiments, two channel analysis (Cy5 vs Cy3 labeled probes), and duplicates versus pooled samples. Control / treatment pairs can be handled normally as long as their normalized values are paired together prior to running the ratio step of the analysis. Time course experiments may require that multiple copies of single or replicate measures of the control timepoint (i.e. - timepoint zero) be made and each paired with the desired experimental time point for the ratio step. Two channel analysis can be treated the same as if each channel was printed on separate arrays, assuming that the chosen normalization method works properly. For two channel analysis, it is recommended that some dye reversal slides (duplicates with the respective dye labels switched) be printed as controls and used in the duplicate comparison step. When there are more than two replicates printed, an average of the replicates can be calculated without filtering for bad duplicates, as this would be pooling the results into one more accurate measure of the gene expression for each treatment. Different yet related samples with the same experimental treatment can be similarly pooled together into an average for each accession.

SUMMARY

Microarray protocols have been written and published on our web page at <http://medir.ohsu.edu/~geneview/> for quality assurance. We then assessed four methods of normalization: mean of standards, 20% trimmed mean slidewise, mean gridwise, and log ratio slidewise. We selected the 20% trimmed mean as the most valid and reproducible method tested, although the log ratio method looked like it might be comparable with

more fine tuning of data processing methods prior to normalization. Fold change ratios give a good estimate of relative gene expression in microarray experiments, but further investigation into statistical methods, possibly using outside applications such as ArrayStat, may provide more accurate results. Comparison of replicates printed on multiple slides are currently our best measure for the reproducibility of microarrays data. The validity of microarray data can be estimated using three measures of slide intensity: the 20% trimmed mean of all spots used during normalization, percentage of spots measured at or above saturation, and the percentage measured below a floor value.

CONCLUSIONS AND FUTURE DIRECTIONS

We found that the 20% slidewise trimmed mean method of normalization was the most reliable of those we tested. Although the slide intensity ratings used for quality control have not been fully tested yet, they appear to be helpful for researchers to help assess the quality of their microarray slides. We have received mostly positive feedback from researchers on the utility of the Automated Normalization program for microarray experiment analysis within our lab.

For our future directions, we would like to port the Automated Normalization program from Microsoft Excel and Visual Basic to Perl or Java running SQL statements directly on our Oracle data warehouse. The advantages to this are speed, efficiency, compatability, and more control over individual data elements. For quality control, we are testing an early implementation of grid alignment indicators in SQL. This is important because we recently found that improper grid alignment during image analysis is one of the largest sources of error in our microarray experiments. The method being tested compares replicates between slides for matches within a given range, giving a percentage of mismatches for each sub-grid. If a particular sub-grid has a high mismatch (low match) percentage, the sub-grid may not be aligned properly, which the researcher can confirm by looking back at the original image and alignment for that grid with Results Reviewer (companion program to AutoGene from BioDiscovery).

BIBLIOGRAPHY

- 1: Bentley DR, *Decoding the Human Genome Sequence*. Human Molecular Genetics, 9(16), pp 2353-2358 (2000)
- 2: Brown P, Botstein D, *Exploring the New World of the Genome with DNA Microarrays*. Nature Genetics, 21(suppl.), pp 33-37 (1999)
- 3: Cheung V.G., Morley M, Aguilar F, Massiami A, Kucherlapati R, Childs G, *Making and Reading Microarrays*. Nature Genetics, 21(suppl.), pp 15-19 (1999)
- 4: Southern E, Mir K, Shchepinov M, *Molecular Interactions on Microarrays*. Nature Genetics, 21(suppl.), pp 5 (1999)
- 5: Schuchhardt J, Beule D, Malik E, Wolski E, Eickhoff H, Lehrach H, Herzel H, *Normalization Strategies for cDNA Microarrays*. Nucleic Acids Research, 28(10), pp e47 (2000)
- 6: Yang Y.H., Dudoit S, Luu P, Speed T.P., *Normalization for cDNA Microarray Data*. SPIE BiOS, (), pp (2001)
- 7: Newton M.A., Kendzierski C.M., Richmond C.S., Blattner F.R., Tsui K.W., *On Differential Variability of Expression Ratios: Improving Statistical Inference about Gene Expression Changes from Microarray Data*. Journal of Computational Biology, 8(1), pp 37-52 (2001)
- 8: Bassett Jr D.E., Eisen M.B, Boguski M.S., *Gene Expression Informatics - It's All in Your Mine*. Nature Genetics, 21(suppl.), pp 51-55 (1999)
- 9: Richmond C.S., Glasner J.D., Mau R, Jin H, Blattner F.R., *Genome-wide Expression Profiling in Escherichia coli K-12*. Nucleic Acids Research, 27(19), pp 3821-3835 (1999)
- 10: Delenstarr G, Cattell H, Chen C, Dorsel A, Kincaid R.H., Nguyen K, et al., *Estimation of the Confidence Limits of Oligonucleotide Array-based Measurements of Differential Expression*. "Microarrays: Optical Technologies and Informatics", Bittner M, et al. Eds., Proceedings of SPIE 4266 in press (2001)

APPENDIX A – DATA DICTIONARY

STAGES OF ANALYSIS:

NAME: DNA / Probe Preparation
DATA ELEMENTS: Cy3, Cy5, label, primer, probe
SOURCE: Microarray experiment design, PCR, mRNA extraction
DESTINATION: Hybridization stage

NAME: Microarray (Target Spot) Printing
DATA ELEMENTS: accession number, slide, target spot
SOURCE: cDNA, clone library, microarray printer, meta-grid
DESTINATION: Hybridization stage

NAME: Hybridization
DATA ELEMENTS: cDNA, clone library, Cy3, Cy5, label, meta-grid, primer, probe
SOURCE: DNA / Probe Preparation stage, Microarray (Target Spot) Printing stage
DESTINATION: Image Scanning stage

NAME: Image Scanning
DATA ELEMENTS: background, gain, PMT, (target) spot intensity
SOURCE: CCD confocal microscope, Hybridization stage
DESTINATION: BArry Microarray Data Warehouse, Software Analysis stage

NAME: Software Analysis
DATA ELEMENTS: accession number, Array_ID, background, (log) difference, duplicates, Experiment_ID, (duplicate / Const_FC) flags, gene names or ESTs, local corrected, meta-grid, norm(alized) value, position, (fold change) ratios, (target) spot intensity, selected, (20%) trimmed mean
SOURCE: AutoGene™ 3.0, Automated Normalization, BArry Microarray Data Warehouse, Image Scanning stage
DESTINATION: cluster analysis (GenExplore™), researcher, scientific discovery

Charged Coupled Device

SOFTWARE ANALYSIS:

NAME: accession number
ALIASES: accession
DESCRIPTION: Identifies cDNA (target spots) printed on microarray slides

DATA TYPE:	character string
LENGTH:	30 characters or less
VALUE:	<p>An accession number is one of the following:</p> <ol style="list-style-type: none"> GenBank accession number: a valid value from NCBI's GenBank database or the ResGen database (obtained from Research Genetics). There are two possible cases: <ul style="list-style-type: none"> The accession number is clustered by UniGene (NCBI) from the web site http://www.ncbi.nlm.nih.gov/UniGene/ or from the BAr ry Data Warehouse table Gene_Accession_Numbers. Most common case. Includes unknown genes or ESTs. The accession number is not clustered by UniGene and obtained from ResGen database as well as GenBank. The string "Standard" followed by a GenBank accession number. Intended for use as "housekeeping" genes (genes with little change in expression between treatments) for some normalization schemes. The string "NA", meaning no cDNA was printed on the target spot. Other identifying characters specific to the microarray facility.
NAME:	Array_ID
ALIASES:	array, channel, image, slide
DESCRIPTION:	<p>Unique identifier for each array which has been gridded, saved, and imported from AutoGene™ to the BAr ry Microarray Data Warehouse. Grouped by Experiment_ID.</p>
DATA TYPE:	character string
LENGTH:	256 characters or less
VALUE:	<p>Not fully standardized, but follows this general format, from left to right:</p> <ol style="list-style-type: none"> First character: <ul style="list-style-type: none"> H = Human genome M = Mouse genome One or two digit number: <ul style="list-style-type: none"> Series number – links meta-grid format with accession numbers on array Dash "-" character delimiter Two digit number: <ul style="list-style-type: none"> Slide number in series Underscore "_" or dash "-" delimiter PMT (integer), followed by a delimiter Gain (integer), followed by a delimiter (optional descriptive string, followed by a delimiter) String "Cy5" or "Cy3" to indicate fluorescent label used (optional delimiter, followed by descriptive string) <p>For example: "H8-02_80_60_Cy5_2nd wash"</p>

NAME: background
 ALIASES: local background, noise
 DESCRIPTION: Fluorescence detected on the microarray slide outside of target spots where no hybridization should occur. Reduced by proper washing of slide after hybridization. Increases or decreases with PMT and gain settings during Image Scanning stage.
 DATA TYPE: Integer (double [64-bit floating point] after computing mean, median, or mode)
 RANGE: 0 to 65536 (2^{16})
 VALUE: BArry stores mean, median, and mode calculations for all background pixels local to each target spot on the array. Median background is typically used for analysis.
 EXCEPTION: AutoGene™ sometimes returns an error string of "NaN" (meaning "Not a Number") when calculating local background. The BArry Microarray Data Warehouse replaces "NaN" with an empty or null value before importing AutoGene™ data.

NAME: difference
 ALIASES: log difference
 DESCRIPTION: Subtraction of log transformed data values for normalization. This is the log transform equivalent of arithmetic ratios, since:

$$\log A - \log B = \log \left(\frac{A}{B} \right)$$

DATA TYPE: Double (64-bit floating point)
 RANGE: 4.82 to -3.00 (log [65536] to log [0.001])
 VALUE: As of 5/30/01, we are testing log difference for normalization as follows:
 $\log(\text{local corrected target spot}) - \log(\text{local corrected all array spots})$

NAME: duplicates
 ALIASES: replicates
 SEE ALSO: flag, floor
 DESCRIPTION: Many microarray experiments will have two or more arrays (slides) printed with identical accession numbers and treatments for repeat measurement of expression. These identical arrays (slides) are referred to as duplicates.
 DATA TYPES: boolean quality indicator flag; mean normalized duplicates
 VALIDITY CHECK: Defined by user during analysis. Default for fold change (ratio) comparisons is 3.00, meaning duplicates with more than 3-fold difference in expression are flagged as bad spots.
 VALUE: Duplicate comparisons are based on the following pseudo-code after setting a floor: $\text{Ratio} = \frac{\text{Max}(\text{duplicate 1}, \text{duplicate 2})}{\text{Min}(\text{duplicate 1}, \text{duplicate 2})}$
Default Duplicate Fold Change = 3.00

If Ratio > Duplicate Fold Change

Then Duplicate Flag = Bad

Else Duplicate Flag = Good

The duplicate flag will be set as follows: *True* means the duplicates are *good*, *False* means the duplicates are *bad*. If the duplicate flag is *True* (*good*), then the normalized values of the two duplicates are averaged. When more than two (2) duplicates (replicates) are available, can simply calculate mean of all duplicates (replicates) without flagging.

NAME: Const_FC
ALIASES: constant fold change
SEE ALSO: flag
DESCRIPTION: A quality indicator flag to test whether a particular accession changes expression beyond a given fold change across the entire experiment set. This can be used to filter expression results by accessions whose expression never changes in the entire experiment.
DATA TYPE: boolean quality indicator flag
VALUE: Default value of Const_FC = 2.00
Flag is *True* if:
$$\frac{\text{maximum normalized value of accession across all arrays}}{\text{minimum normalized value of accession across all arrays}} > \text{Const FC}$$

otherwise *False*.

NAME: Experiment_ID
DESCRIPTION: Identifier for independent microarray experiments. Associates the arrays (slides) printed with a given experiment.
DATA TYPE: character string
VALIDITY CHECK: Should be a valid Windows 9x/NT file or folder name
VALUE: Not standardized at this time. Recommend name of experiment or name of person performing the experiment, a *space* character, then the date the data analysis was performed in *MM-DD-YY* format.

NAME: flag
DESCRIPTION: A boolean or character indicator for filtering and/or quality control.
DATA TYPE: boolean or character
VALUE: boolean: *True* or *False*; generally *True* = *good*, *False* = *bad*
character: varies

NAME: ESTs
SUBSET OF: accession numbers
DESCRIPTION: EST – Expressed Sequence Tags
VALUE: Can be filtered out of experiment results as unknown sequences, or linked to known genes during cluster analysis.

NAME:	floor
DESCRIPTION:	A set minimum value for a given data element. If the given data element is less than the floor value, the data element is set equal to the floor value, thus restricting the low end of its range to a minimum value.
DATA TYPE:	double (64-bit floating point)
VALUE:	User defined. For duplicate comparisons, the default floor value is 0.1
NAME:	fold change
SEE ALSO:	ratios, Const_FC, duplicates
DESCRIPTION:	Narrows the definition of ratios to describe relative gene expression.
DATA TYPE:	Double (64-bit floating point)
RANGE:	0 to 65536 (2 ¹⁶)
VALUE:	Default fold change to determine differential expression is 3.00 Default fold change to determine Const_FC is 2.00 Narrows the definition of ratios to read as follows: If <i>ratio</i> > 1, then the expression of <i>X</i> is <i>fold change</i> times (fold) greater than <i>Y</i> . If <i>ratio</i> < 1, then the expression of <i>X</i> is $\frac{1}{\text{fold change}}$ times (fold) less than <i>Y</i> .
NAME:	gene name
SEE ALSO:	accession number
DESCRIPTION:	Descriptive name of accessions obtained from current GenBank or ResGen databases, linked to accession number.
DATA TYPE:	character string
LENGTH:	100 characters or less
VALUE:	Names of known genes matching given accession number in UniGene or ResGen databases.
NAME:	intensity
ALIAS:	signal
DESCRIPTION:	The amount of fluorescence detected from laser scanning using one or more pixels of the resulting image on the computer.
DATA TYPE:	integer (double [64-bit floating point] after computing mean, median, or mode)
RANGE:	65536 (2 ¹⁶ or 16-bit grayscale)
VALUE:	The mean, median, or mode of all pixels in a desired region (i.e. - target spot or local background).
NAME:	local corrected
DESCRIPTION:	The target spot corrected by removal of the local background.

DATA TYPE: double (64-bit floating point)
RANGE: 0 to 65536 (2^{16})
VALUE: Calculated using the following formula:

$$\text{target spot intensity} - \text{local background intensity}$$

NAME: log ratio
SEE ALSO: difference
DESCRIPTION: A normalization method using log transformation of local corrected data.
DATA TYPE: double (64-bit floating point)
RANGE: 4.82 to -3.00 (log [65536] to log [0.001])
VALUE: As of 5/30/01, we are testing difference of logs for log ratio normalization as follows:

$$\log(\text{local corrected target spot}) - \log(\text{local corrected all array spots})$$

NAME: meta-grid
SEE ALSO: meta-row, meta-col, sub-grid, position
DESCRIPTION: Coordinate location of two-dimensional sub-grids printed on the original array (slide). Used to track the position of accessions on the microarray.
DATA TYPE: integer
RANGE: 1001 to 998001 (999 x 999 grid) in position notation
VALUE: In position notation, the first one to three digits without leading zeros are the meta-row number from top to bottom. The next three digits are the meta-col number from left to right. Refer to *position* for more information.

NAME: meta-col
SEE ALSO: meta-grid, meta-row, sub-grid, position
DESCRIPTION: Column number of one sub-grid within a two-dimensional arrangement of sub-grids.
DATA TYPE: position notation or integer
RANGE: Position notation: 001 to 999 (three digits with leading zeros)
 Integer: 1 to 999
VALUE: Column number of sub-grids from left to right, as printed on original array.

NAME: meta-row
SEE ALSO: meta-grid, meta-col, sub-grid, position
DESCRIPTION: Row number of one sub-grid within a two-dimensional arrangement of sub-grids.
DATA TYPE: position notation or integer
RANGE: Position notation: 1 to 999 (one to three digits without leading zeros)
 Integer: 1 to 999
VALUE: Row number of sub-grids from top to bottom, as printed on original array.

NAME: norm(alized) value

SEE ALSO:	(20%) trimmed mean, log ratio
DESCRIPTION:	The relative intensity of each target spot, corrected for systematic variation of arrays (slides) across entire experiment set.
DATA TYPE:	double (64-bit floating point)
RANGE:	0 to 65536 (2^{16})
VALUE:	Dependent on the chosen normalization method for data analysis.
NAME:	position
SEE ALSO:	meta-grid, meta-row, meta-col, sub-grid, sub-row, sub-col
DESCRIPTION:	A notation to describe the absolute location of each target spot (accession) on a given array (slide). Used to sort by grid location and locate the original spot on the array.
DATA TYPE:	position notation (derivative of integer type)
RANGE:	1001001001 to 999999999999
VALUE:	Concatenation of 10 to 12 digits as follows: First one to three digits without leading zeros = meta-row Next three digits with leading zeros = meta-col Next three digits with leading zeros = sub-row Last three digits with leading zeros = sub-col
NAME:	ratio
SEE ALSO:	fold change
DESCRIPTION:	Method of determining relative expression by dividing the norm value of one experimental condition by another.
DATA TYPE:	double (64-bit floating point)
RANGE:	0.001 to 65536 (2^{16})
VALUE:	
NAME:	spot
ALIASES:	target, target spot
DESCRIPTION:	Circular area of cDNA printed on a microarray slide from a clone library. This is the region of the microarray slide where hybridization of a matching fluorescent labeled probe to the spotted cDNA will occur. After image analysis, this area is represented as a circular group of pixels, detected and intensity computed using AutoGene™ software from BioDiscovery.
NAME:	selected
DESCRIPTION:	A flag that indicates whether a spot was marked as bad by the technician using AutoGene™ from BioDiscovery during the image analysis process or given an error value such as "NaN" (Not a Number) when imported to the BArray Microarray Data Warehouse.
DATA TYPE:	integer
RANGE:	binary

VALUE: 0 = bad (deselected), 1 = good (selected)

NAME: sub-col
 SEE ALSO: sub-grid, sub-row, meta-grid, position
 DESCRIPTION: Column number of one spot in a two-dimensional sub-grid.
 DATA TYPE: position notation or integer
 RANGE: Position notation: 001 to 999 (three digits with leading zeros)
 Integer: 1 to 999
 VALUE: Column number within a sub-grid from left to right, as printed on original array.

NAME: sub-grid
 SEE ALSO: sub-row, sub-col, meta-grid, position
 DESCRIPTION: Coordinate location of spot printed in a two-dimensional arrangement on the original array (slide) within a given meta-grid. Used to track the position of accessions on the microarray.
 DATA TYPE: position notation
 RANGE: 001001 to 999999 (999 x 999 grid) in position notation
 VALUE: In position notation, the first three digits with leading zeros are the sub-row number from top to bottom. The next three digits with leading zeros are the sub-col number from left to right. Refer to *position* for more information.

NAME: sub-row
 SEE ALSO: meta-grid, meta-col, sub-grid, position
 DESCRIPTION: Row number of one spot in a two-dimensional sub-grid.
 DATA TYPE: position notation or integer
 RANGE: Position notation: 001 to 999 (one to three digits with leading zeros)
 Integer: 1 to 999
 VALUE: Row number within a sub-grid from top to bottom, as printed on original array.

NAME: (20%) trimmed mean
 DESCRIPTION: A normalization method which removes outliers before computing the mean of all spots (accessions) on the array (slide)
 DATA TYPE: double (64-bit floating point)
 RANGE: 0 to 65536 (2^{16})
 VALUE: Remove top 10% and bottom 10% of all local corrected values on the slide, then compute the mean of the remainder. Divide the local corrected value for each accession on the slide by the trimmed mean.

APPENDIX B – MICROARRAY PREPARATION PROTOCOLS

DNA Preparation of Clones for Printing: Amine Modification of DNA & Polymerase Chain Reaction (PCR) Protocol

Use Amine C-12 Primers:

GF 200 Forward: C-12 Amine 5' CTGCAAGGCGATTAAGTTGGGTAAC 3'

GF 200 Reverse: C-12 Amine 5'

GTGAGCGGATAACAATTTACACAGGAAACAGC 3'

This protocol calls for a 100 λ (100 μ l) total reaction volume.

Our typical "master mix" formula would be:	<u>Rxn.</u>	Volume for 4-96 well <u>plates</u>	Final volume <u>per indiv. well</u>
[500nM] final concentration forward primer	5 λ	2060 μ l	5 μ l
[500nM] final concentration reverse primer (Diluted from 10 μ M stock)	5 λ	2060 μ l	5 μ l
10x Magnesium free buffer	10 λ	4120 μ l	10 μ l
MgCl ₂ (add to select desired specificity) (Diluted from 25mM stock)	5 λ	2060 μ l	5 μ l
[200 μ M] final concentration dNTP's <i>Amersham Pharmacia Biotech™</i> (Product # 21-2094) (Diluted from 20mM stock)	1 λ	412 μ l	1 μ l
Taq polymerase (added last) (Reaction Buffer, MgCl ₂ & Taq polymerase from Promega Catalog #M1865)	0.5 λ	206 μ l	0.5 μ l
DEPC H ₂ O (added to bring to target volume)	<u>x λ</u>	<u>29,458μl</u>	<u>71.5μl</u>
Total Volume:	98λ	40,376μl	98μl

PCR checklist:

Multiply each of the above by the total number of reactions required.

**Remember to add Taq polymerase last.*

Aliquot each 98µl reaction to 100 PCR reaction tube.

Add 2µl of overnight culture/glycerol stock ("stabs") or "housekeeping gene" to equal **100µl total**.

Keep track of loading order; label if necessary.

Load into DNA engine and run with the following cycle protocol:

Step 1: 96°C for 30 seconds

Step 2: 94°C for 45 seconds

Step 3: 55°C for 45 seconds

Step 4: 72°C for 2 minutes & 30 seconds

Step 5: Goto 2, **35 times**

Step 6: 72°C for 5 minutes

Step 7: 4°C forever

Step 8: End

**Use Heated Lid Option*

PCR Clean Up (Array It™ protocol):

1. Spin down plates from PCR reactions to insure entire volume is in apex of tube.
2. Position Super Filter 100 on 96-well vacuum manifold.
3. Add 500µl of **Array It Binding Buffer** to each well of the Super Filter 100 using a 12-channel pipetting device set for 500µl and should be performed as quickly as possible (preferably 1' per plate) to minimize loss of Binding Buffer due to gravity flow. Be careful to not splash contents between wells.
4. Quickly add 100µl per well of PCR sample for a 96 well plate to the corresponding well of the Super Filter 100; **very crucial for later database analysis**. Transfer should take place as quickly as possible (<1') to prevent buffer loss due to gravity.
5. Immediately mix the Binding Buffer and the PCR sample thoroughly by pipetting up and down 10 times with the 12 well pipetting device. Mixing should once again be completed ASAP (<5') to limit loss of Buffer due to

gravity. ***Special care should be taken to insure that wells are not splashed when adding and mixing PCR product***

6. Apply a gentle vacuum to the Super Filter 100 to allow binding of the PCR product to the Super Filter 100 membrane. Primers, nucleotides, single-stranded products, salts, and other impurities pass through the Super Filter 100 into the waste reservoir at the bottom of the vacuum filtration apparatus.
7. Shut off the vacuum and add 500µl of Wash Buffer to each well of the Super Filter 100 with a 12-channel pipetting device. Apply gentle vacuum until Wash Buffer has passed through the membrane; repeat this process again with another 500µl Wash Buffer. These two washes are used to remove Binding Buffer and PCR sample that has adhered to the wall of the wells during mixing.
8. Repeat wash once again with 100µl of Wash Buffer; Apply vacuum until membrane is dry (~3-4'); this will insure trace contaminants and Wash Buffer are removed and the membrane bound PCR product is fixed to the membrane.
9. Remove the Super Filter 100 from the vacuum manifold and place it on an *"unmarked"* 96-well plate.
10. Centrifuge the two plates for 5' at ambient temp @ 500g (600rpm) to remove small amounts of Wash Buffer. Discard the unmarked microplate containing the residual Wash Buffer.
11. Transfer the Super Filter 100 containing the bound PCR product onto a *"marked"* 96-well microplate.
12. Rehydrate the Array It Super Filter by adding 100µl per well of DEPC H₂O with automatic pipetting device (time not critical here); for maximal recovery of DNA be sure to add DEPC H₂O directly to the surface of the Super Filter membrane.

Preparation of Printed Slides for Hybridization & Scanning

Silylated slides will be used for printing microarray (TeleChem. Intl. Superaldehyde CSS-100, CEL Associates, Inc.).

Processing of silylated slides after printing:

Schiff Base Attachment:

1. Slides are washed once in a 0.2% SDS immersion for 1'
 2. Rinse twice in ddH₂O for 1' each time
 3. Treat in sodium *borohydride* solution, make fresh each time (1.0g NaBH₄ in 300mL of PBS and 100mL of 100% EtOH) for 5'.
 4. Rinse slides once in ddH₂O for 2' at 95°C.
 5. Rinse once in 0.2% SDS for 1'.
 6. Rinse twice in ddH₂O, air dry, and store in the dark at RT.
-

Fluorescent Labeling Probes:

INDIRECT, BIOTINYLATED CDNA SYNTHESIS

Total Reaction volume is 20 λ (20 μ l)

*5 μ g of total RNA or 1 μ g of poly A+ from desired source. Prepared using CsCl₂, Qiagen, or Trizol methods.

Basic schematic of both Indirect and Direct labeling (below) is the same, as such:

RNA/Oligo/ H₂O:

RNA [5 μ g total or 1 μ g poly A+]

5 λ

Oligo dT primer (24) [1μg/μl]	2λ
N-1	1λ
H ₂ O (to volume)	<u>xλ</u>
Total:	7.9λ

Reaction Buffer:

5x Buffer (GIBCO-BRL ; see below):	4λ
0.1M DTT [10mM final]	2λ
10mM dNTP's (AGT) [300μm final]	0.6λ
2 mM dCTP "cold" (unlabeled) [150μm final]	1.5λ
dCTP-Biotin-11 [150μm final]	3λ
Superscript™ II (Reverse Transcriptase) (200 units)	<u>1λ</u>
<i>Added absolutely last to each tube; starts reaction.</i>	
Total:	12.1λ

Plus above = (7.9λ) = 20.0λ

1. Place the RNA/Oligo/ H₂O mixture in 65°C water bath for 10' and then immediately on ice for 2'.
2. Combine RNA/Oligo/ H₂O tube with Reaction Buffer
3. Allow Reverse Transcription to proceed for 60-120' @ 37°C
4. Then @ 45°C for 15' and spin down.
5. Add 180μl of DEPC H₂O, making total volume 200μl.
6. Add 20μl of 3M NaAc, and 400μl of EtOH, and precipitate the product @ -20°C for 30'
7. Spin down for another 30' to pellet product; *discard the supernatant.*
8. Vacuum dry (lyophilize) the pellet.

*Verify purity both via 1% agarose gel and Optical Density (OD) obtained from spectrophotometer (260/280 Ratio should be 1.6 or greater)

First strand buffer:	4μl (1x Buffer)
250mM Tris-HCl	
375m M KCl	
15mM MgCl ₂	
10mM dNTP's (C,G,T):	0.6μl 300μM dNTPs
10μl of dCTP (each as 100mM stock + 70μl dH ₂ O)	
10μl of dGTP	
10μl of dTTP	

N-1 is an internal RNA control for labeling efficiency.

Superscript II Catalog #18064-014 Also Contains 5x First Strand Buffer

DIRECT LABELING, CDNA SYNTHESIS (LABEL DIRECTLY INCORPORATED INTO CDNA)

Total Reaction volume is 20λ (20μl)

2μg of total RNA from desired source.

Basic schematic is such:

RNA (1-2μg poly A+ or 5μg total)	xλ	1-5μl
Oligo dT primer (24) (50pmol)	2λ	2μl
N-1 (for "spiking" reaction)	1λ	1μl
H ₂ O (to volume)	<u>xλ</u>	<u>xμl</u>
Total	λ	

Place the above mixture in 65°C water bath for 10' and then immediately on ice for 2'.

While the above is incubating; Add the following to a clean 1.5mL tube:

5x Buffer (GIBCO-BRL ; see below):	4λ	4μl
0.1M DTT [10mM final]	2λ	2μl

10mM dNTP's (AGT) [300µm final]	0.6λ	0.6µl
2 mM dCTP cold [150µm final]	1.5λ	1.5µl
dCTP-Cyanine 3 or 5 [150µm final]	3λ	3.0µl
Superscript II	1λ	1.0µl
Total:	12.1	
Total Volume		20µl

1. Add each of the above reactants in the order in which they appear; allow Reverse Transcription to proceed for 60-120' @ 37°C

2. Then @ 45°C for 15' and spin down (optional)

3. Add 180µl of DEPC H₂O, making total volume 200µl.

4. Add 20µl of 3M NaAc, and 400µl of EtOH, and precipitate the product @ -20°C for 30'

5. Spin down for another 30' to pellet product; *discard the supernatant*.

6. Vacuum dry (lyophilize) the pellet

1-2µg of poly A+ RNA, verify purity both via 1%agarose gel and Optical Density (OD) obtained from spectrophotometer (260/280 ratio should be 1.6 or greater)

N-1 is an internal RNA control for labeling efficiency

Superscript II Catalog #18064-014 Also Contains 5x First Strand Buffer

Hybridization:

1. Re-suspend purified, labeled probe cDNA in DEPC H₂O to either:
10µl DEPC H₂O if Slide/Cover slip will be utilized for hybridization, or
30µl DEPC H₂O if *Perfusion Chamber* is used (**CoverWell™** Grace Bio-Labs,

Catalog # PC1L-0.5)

2. Boil probe for 5' in hot water bath to ensure denaturation.
- 2
3. Heat Ribohybe (see recipe's below); to 50°C
4. Add entire volume of suspended probe to either 70µl Ribohybe (for slide/cover); or 300µl Ribohybe if slide chamber will be used for hybridization.
5. Apply solution to slide/cover or slide/chamber and place in humidifying hybridization chamber; place in incubator @ 50°C overnight (8 to 24 hours).

Recipe's for above:

50x Denhardt's (stored @ -20°C)
 10g Ficoll 400
 10g Polyvinylpyrrolidone (PVP)
 10g Bovine Serum Albumin Fraction V (BSA) (stored @ -4°C)
 Bring to 1L volume/aliquot into 100mL, store @ -20°C

Ribohybe solution; for 1L:
 500mL Formamide (extremely volatile)
 250mL 20x SSC
 100mL 50x Denhardt's (stored @ -20°C)
 50mL 0.5M Phosphate buffer, pH 7.0
 50mL SS DNA 10mg/mL (stored @ -20°C)
 25mL yeast tRNA (stored @ -20°C)
 50g SDS (use mask/goggles)
 Warm gently while stirring, add SDS last don't heat over 50°C (volatile), pH 7.4
 Aliquot into 300-400mL in 500mL bottles. Store @ -20°C.

PREPARATION OF SLIDES FOR SCANNING

Protocol modified from the *Renaissance® TSA™ Direct (Cyanine 3)*
(Tyramide Signal Amplification) Kit by New England Nuclear **Catalog #**
NEL704A

Reagents also available separately.

13. Quick wash slides to remove probe with 2x SSC (dilute from 20x) for 1' two times and once on rocker for 5'.
2. Heat 0.2x SSC, to 50°C and quick wash slides twice and then once more on rocker for 5'.
3. ***(See Below)** Block slides for 30' in 500µl **TNB** buffer @ RT; being careful not to let any section of the slide dry out-check periodically.
4. Add 500µl Streptavidin-HRP (kit) diluted 1:100 in **TNB** buffer (5µl Strept-HRP conjugate + 495µl TNB) to slide @RT for 30'
5. Wash 3x in **TNT** buffer for 5' each wash
6. Add Cyanine 3-Tyramide conjugate to slides (6µl stock + 300µl amplification/diluent buffer) for 10'.
7. Wash 3x for 5' with **TNT** buffer @ RT
8. Rinse with 0.2x SSC for 2'.
9. Centrifuge @ 800rpm for 2-5' to thoroughly dry slide.

Slide is now ready to scan; take to Microarray lab.

Recipe's for above:

TNB buffer:

0.1M Tris-HCl, pH 7.5
0.15M NaCl
0.5% Blocking Reagent (supplied in Kit)

TNT buffer:

0.1M Tris-HCl, pH 7.5
0.15M NaCl
0.05% Tween 20
Sterilize with 0.22 μ m vacuum filter

20x SSC:

For 4L,
701.4g NaCl
353.0g Citric Acid

For Direct Labeling; do steps 1&2 only, followed by drying and scanning as usual.

REVERSE TRANSCRIPTION PROTOCOL

Isolate RNA using Qiagen-RNAEasy kit, Cesium Chloride, or Trisol.

Reverse Transcription

Steps:

1. Total RNA = 1 or 2 μg

Oligo dT Primer = 25 pmoles

RT Cocktail-To be added last (See Below) = 8 μL

H₂O = bring up to 20 μL

RT Cocktail per reaction (Gibco Superscript II Kit is recommended)

5X 1st Strand Buffer = 4 μL

0.1 mM DTT = 2 μL

10 mM dNTP = 1 μL

Superscript II Enzyme = 1 μL

2. Add H₂O, then RNA, and Oligo dT primer, and let stand at room

temperature while preparing the RT cocktail.

3. Add RT Cocktail to the tubes containing the H₂O, RNA and Oligo dT

primer.

4. Cap the tubes securely, and place in a PCR Block / Thermal Cycler

for the following program:

a. 37° C for 60 minutes

b. 45° C for 15 minutes

c. 95° C for 5 minutes

d. (Optional) 4° C until removal
from PCR Block

PREPARATION OF RT PRODUCT FOR QUANTITATIVE PCR

1. Prepare 10 μ M, 100 μ L stocks of the 7700 forward and reverse primers,
and the corresponding probes. The Taqman primers and probe can be designed
using the computer program, ABI Primer Express.
2. Spin the RT Tubes at 1500 to 2000 rpm at 4 degrees Celsius for 5
minutes to collect the RT product. Spec the cDNA at 260 nm to
determine its concentration and dilute 5 μ L of the cDNA in H2O to a
final concentration of 20 ng/ μ L.
3. The cDNA samples and the respective primers and probes are now ready
to undergo quantitative PCR and should be stored separately
to prevent contamination of the primers and probes.

QUANTITATIVE PCR PROTOCOL

Reagents per reaction:

cDNA = 100 ng , 5 μ L

Gene of Interest Forward Primer (10 μ M) = 0.75 μ L

Gene of Interest Reverse Primer (10 μ M) = 0.75 μ L

Gene of Interest Probe (10 μ M) = 0.25 μ L

Housekeeping Forward Primer (10 μ M) = 0.25 μ L

Housekeeping Reverse Primer (10 μ M) = 0.25 μ L

Housekeeping Probe (10 μ M) = 0.25 μ L

Applied Biosystems PCR Master Mix = 12.5 μ L

Master Mix Components

10X TaqMan Buffer A

25 mM MgCl₂ Solution

dATP, dCTP, dGTP, dUTP

AmpliTaq Gold

AmpErase UNG

H₂O = 5.0 μ L

Quantitative PCR :

Run ABI 7700 Sequence Detector

1. 50° C for 2 minutes
2. 95° C for 10 minutes
3. Data Collection Stage
 - a. 95° C for 15 seconds
 - b. 60° C for 1 minute
 - c. 40 cycles of steps 3a and 3b
4. 4° C until removal from Sequence Detector

QUANTITATIVE PCR DATA COLLECTION AND ANALYSIS

The ABI PRISM 7700 Sequence Detection System integrates a PCR-based assay and hardware/software instrumentation for high-through-put quantitation of nucleic acid sequences. Quantitative detection of specific nucleic acid sequences is possible using the fluorogenic 5' nuclease assay. With this chemistry, a fluorogenic probe complementary to the target sequence is added to the PCR reaction mixture. The probe consists of an oligonucleotide with a reporter and quencher dye attached. During PCR, if the target of interest is present, the probe anneals specifically between the forward and reverse primer sites. The nucleolytic activity of the polymerase cleaves the probe, which results in an increase in the fluorescence intensity of the reporter dye. To induce fluorescence during PCR, laser light is distributed to the sample wells via a multiplexed array of optical fibers. The resulting fluorescent emission returns via the fibers and is directed to a spectrograph with a charge-coupled device (CCD) camera.

For each sample, the CCD camera collects the emission data between 520 nm and 660 nm once every few seconds. The software analyzes the data by first calculating the contribution of each component dye to the experimental spectrum. The reporter signal is then normalized to the fluorescence of an internal reference dye. Peak normalized reporter values are averaged for each cycle and plotted versus cycle number to produce an amplification plot.

The parameter, CT (threshold cycle), is defined as the fractional cycle number at which the reporter fluorescence generated by cleavage of the probe passes a fixed threshold above baseline. A plot of the log of initial target copy number for a set of standards versus CT is a straight line. Quantitation of the amount of target in unknown samples is accomplished by measuring CT and using the standard curve to determine starting copy number. The entire process of calculating CTs, preparing a standard curve, and determining starting copy number for unknowns is performed by the software of the 7700 system.

Relative gene expression can also be determined based on the threshold cycles of the gene of interest and of the internal reference gene. Use of a reference or housekeeping gene accounts for differences in starting cDNA. Subtraction of the CT of the housekeeping gene from the CT of the gene of interest yields:

$$\text{CT Gene of interest} - \text{CT Housekeeping} = \Delta\text{CT}$$

The Change in CT, ΔCT , can then be normalized :

$$\Delta\text{CT sample} - \Delta\text{CT calibrator} = \Delta\Delta\text{CT}$$

Utilizing the normalized value, $\Delta\Delta\text{CT}$, comparative fold values can be calculated:

$$\text{Relative Quantification} = 2^{-\Delta\Delta\text{CT}}$$

Quantification of dsPCR product protocol for Bio-Rad Fluoromark Readers

Read mode: multi-plate

Wavelengths: excitation: 485nm, emission: 538nm

Data display Option:

RFU

To set up standard curve

make a 2 µg/ml lambda DNA stock in TE buffer

[final] ng/ml	TE(µl)	lambda DNA stock
1	999	1 µl
10	990	10 µl
50	950	50 µl
100	900	100 µl
200	800	200 µl

Dilution of PicoGreen dye:

100 µl of PicoGreen stock solution makes 20 ml working solution.

Add 100 µl of PicoGreen stock solution into 19.9 ml TE buffer. Wrap tube with foil to prevent exposure to light.

PCR products dilution:

1. Add 198 µl of TE to 96-well plate, then add 2 µl of PCR product to the 96-well plate, mix well by pipetting up and down. (this is 100x dilution)
2. Add 90 µl of TE buffer to black 96-well plate. take 10 µl of diluted PCR

product into black assay plate, mix well by pipetting up and down.
(This is 10x dilution)

3. Add 100 μ l of diluted PicoGreen solution into each 96-well plate, mix well. (This is 2x dilution, so final dilution is 2000x)

4. Incubate at room temperature for 2-5 min then read on fluorometer.

(For reading multiple plates check the reference of Microplate Manager III software manual page 11-14)

Use Bio-Rad Fluoromark Readers

1. double click **PROT dsDNA mult** on Mac computer to open the program
2. Choose **Read Multiple Plates** from the Plate menu
the reader setup dialog box will appear
3. Choose **Reader Setup** conditions and click **OK**
before each plate is read, a dialog box will appear, asking you to enter a descriptive title for the plate.
4. Give a title for the first plate and click **OK**.
(The same dialog will appear before each plate)
5. When there are no more plates to read, click **STOP**
6. Enter a description for the set of plates in the Plate Description window.

Under Transformations:

only flag box 1: @-#, #=0 (# is the blank well)

Shake: OFF

Leave Transformed Response box blank

Leave Transformed Unit box blank

7. After reading, go to **Plate ID** select each plate, then go to Analysis menu open **Template and Data** to display results.

APPENDIX C – AUTOMATED NORMALIZATION
PROTOCOL

Automated Normalization Protocol

Srinavasa Nagalla Lab Microarray Core Facility

Written By: Matthew J. Rodland

rodlandm@ohsu.edu

Please report any bugs in the Automated Normalization program to Matthew
Rodland at rodlandm@ohsu.edu

9/25/2001

Automated Normalization Protocol - Table of Contents

Section 0: Preparation

Section 1: Import Raw Text Data from BArny Oracle Data Warehouse to Excel

Section 2: Local correction for background (mean signal – median background)

Section 3: Normalize Excel Data

20% Trimmean all genes by channel / slide (preferred method)

Meta Grid Normalization

Section 4: Analyze Duplicates

Section 5: Calculate Ratios (by slide)

Section 6: Add Gene Name to Accession Number

Section 7: Sort Results by Accession Number

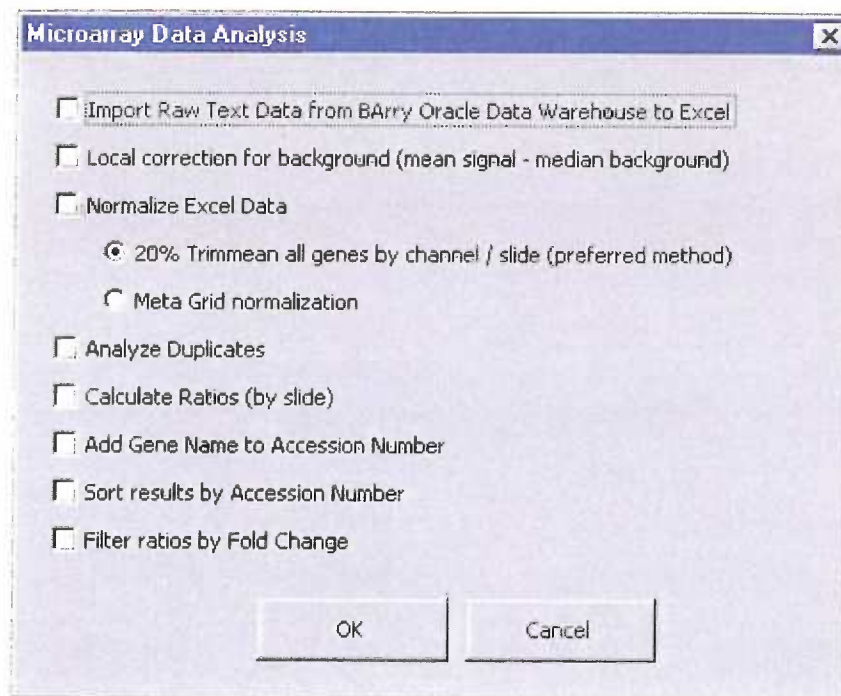
Section 8: Filter Ratios by Fold Change

SECTION 0: PREPARATION

1. Extract raw expression data from BArray Microarray Data Warehouse, noting the location of the saved file. (e.g. - P:\users\yourname\raw_data_example.txt)
2. Run Microsoft Excel (97 or 2000)
3. Select Tools » Macro » Macros... from the Menubar
 - a. Select *Rodland_Macro...* from the **Macros in** drop down menu.
 - b. Select *Automated_Normalization* from **Macro name** list
 - c. Click the **Run** button

Shortcut: Press <CTRL> + <SHIFT> + A on the keyboard

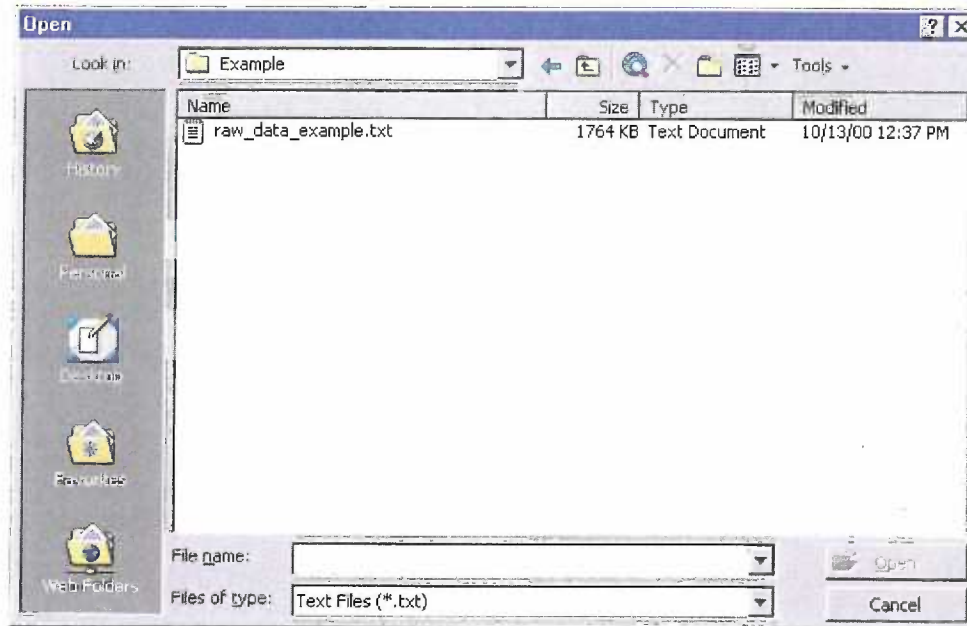
4. The **Microarray Data Analysis** window appears. Each checkbox selects which steps of the Automated Normalization are going to be run. These steps will be run in the order listed on this window, from top to bottom. Each step is described in their own section in this protocol.



SECTION 1: IMPORT RAW TEXT DATA FROM BARRY ORACLE DATA WAREHOUSE TO EXCEL

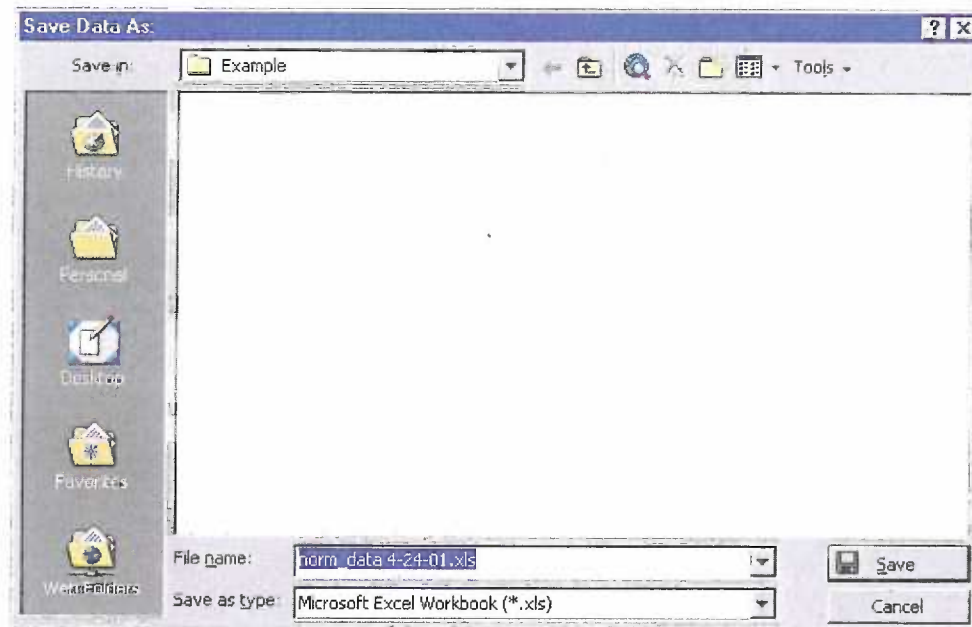
Run the *Automated Normalization* program as described in [Section 0](#) with the checkbox next to *Import Raw Text Data...* selected.

1. **Open** dialog box appears:



2. Browse to the folder where the raw data saved from the BArry Microarray Data Warehouse is located.
3. Select the text file saved from the BArry Microarray Data Warehouse at noted previously (e.g. - browse to P:\users\yourname\raw_data_example.txt), then click the **Open** button.

4. **Save Data As** dialog box appears:



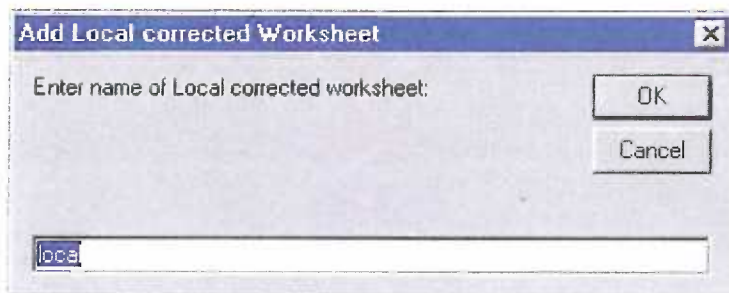
5. Browse to the location you wish to save the normalized data to (e.g. - P:\users\yourname\Example)
6. Type the desired name of the normalized data file in the **File name:** field (or accept the default *norm_data MM-DD-YY* where *MM-DD-YY* is today's date).
7. Make sure the **Save as type:** field reads *Microsoft Excel Workbook (*.xls)*. This file type should be set by default.
8. Click on the **Save** button.

SECTION 2: LOCAL CORRECTION FOR BACKGROUND (MEAN SIGNAL – MEDIAN BACKGROUND)

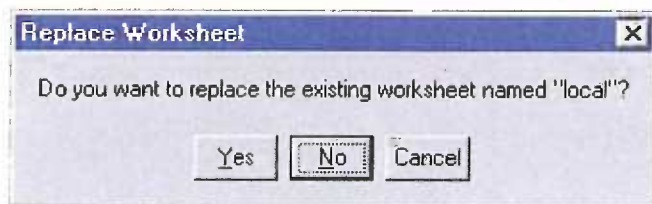
This step in the Automated Normalization assumes that you have the raw text data from the BARRY Microarray Data Warehouse imported to Excel by running the **Import Raw Text Data...** step as described in [Section 1](#). If you have created and saved the spreadsheet previously, then open that spreadsheet now. The raw text import worksheet created in [Section 1](#) should be open in Excel before running this step.

Run the *Automated Normalization* program as described in [Section 0](#) with the checkbox next to *Local correction for background...* selected.

1. Add Local corrected Worksheet dialog box appears:



2. Accept the default name, or type the name of the *worksheet* for calculating and storing the microarray data locally corrected for background. Remember that an Excel *worksheet* is one page of an Excel *workbook*, accessible by clicking on the tab with the given name of the *worksheet* at the bottom of the screen. An Excel *workbook* is the complete Excel file as stored on disk, containing one or more *worksheets*.
3. Click **OK** to continue, or **Cancel** to exit the Automated Normalization.
4. If a worksheet by the same name already exists, you will see the following dialog box, otherwise the **Local correction for background** step is done:



5. Choose from the following:
 - a. Click **Yes** if you wish to overwrite the existing worksheet with the same name (shown in quotes) and replace it with the new data.
 - b. Click **No** if you wish to go back to step 1 and rename the new worksheet.
 - c. Click **Cancel** to stop the Automated Normalization and return to Excel.

SECTION 3: NORMALIZE EXCEL DATA

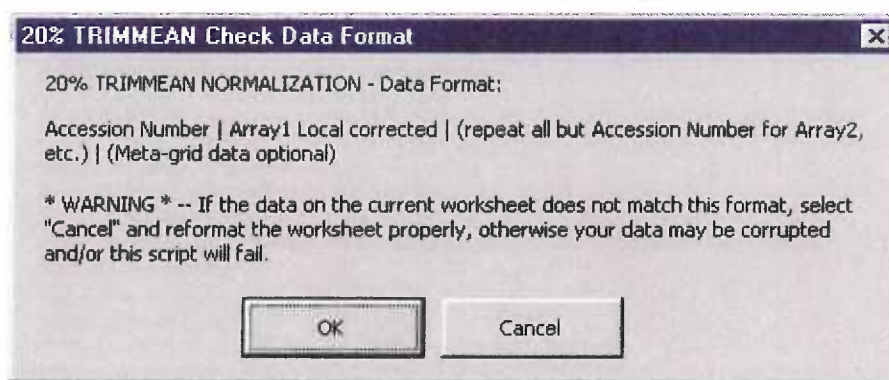
This step in the Automated Normalization assumes that you have the local corrected data worksheet in Excel created by running the **Local correction for background...** step as described in [Section 2](#). If you have created and saved the spreadsheet previously, then open that spreadsheet now. The local corrected worksheet created in [Section 2](#) should be open before running this step.

On the **Microarray Data Analysis** dialog box, you have a choice of normalization methods. Each method will be described separately in the following subsections.

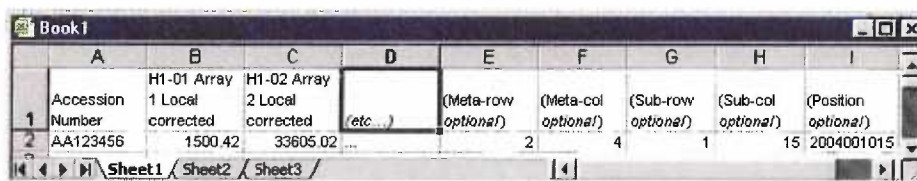
20% TRIMMEAN ALL GENES BY CHANNEL / SLIDE

Run the *Automated Normalization* program as described in [Section 0](#) with the checkbox next to *20% Trimmean all genes...* selected.

1. The **20% Trimmean Check Data Format** dialog box appears:

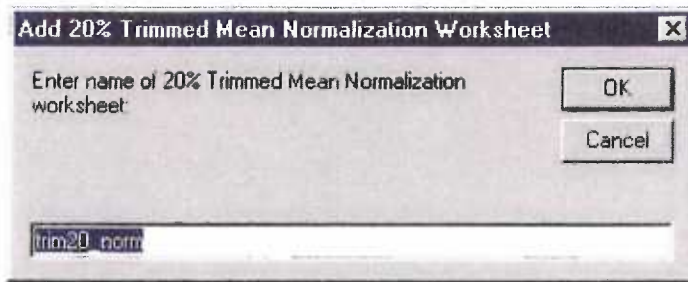


2. Make sure that the open worksheet is in the format described in the dialog box, where the vertical bars delimit each column of the worksheet, from left to right. Follow the dialog boxes instructions. The following illustration shows an example of the proper format for data prior to running the **20% Trimmed Mean Normalization** step:

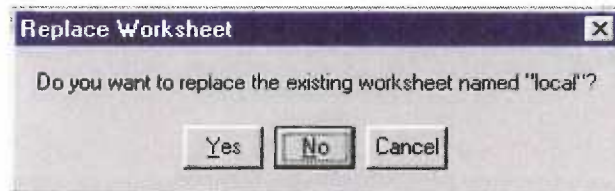


	A	B	C	D	E	F	G	H	I
1	Accession Number	H1-01 Array 1 Local corrected	H1-02 Array 2 Local corrected	(etc...)	(Meta-row optional)	(Meta-col optional)	(Sub-row optional)	(Sub-col optional)	(Position optional)
2	AA123456	1500.42	33605.02	...	2	4	1	15	2004001015

3. If the current worksheet is not in the proper format, click on **Cancel** to exit the Automated Normalization and manually edit the worksheet before running the Automated Normalization program again. Otherwise, click on **OK** to continue.
4. The **Add 20% Trimmed Mean Normalization Worksheet** dialog box appears:



5. Type the name of the worksheet for calculating and storing the normalized data, or accept the default name. Remember that an Excel *worksheet* is one page of an Excel *workbook*, accessible by clicking on the tab with the given name of the *worksheet* at the bottom of the screen. An Excel *workbook* is the complete Excel file as stored on disk, containing one or more *worksheets*.
6. Click on **OK** to continue, or **Cancel** to exit the Automated Normalization.
7. If a worksheet by the same name already exists, you will see the following dialog box, otherwise the **Normalize Excel Data** step is done:



8. Choose from the following:
 - a. Click **Yes** if you wish to delete the existing worksheet with the same name (shown in quotes) and replace it with the new data.
 - b. Click **No** if you wish to go back to step 4 and rename the new worksheet.
 - c. Click **Cancel** to stop the Automated Normalization and return to Excel.

META GRID NORMALIZATION

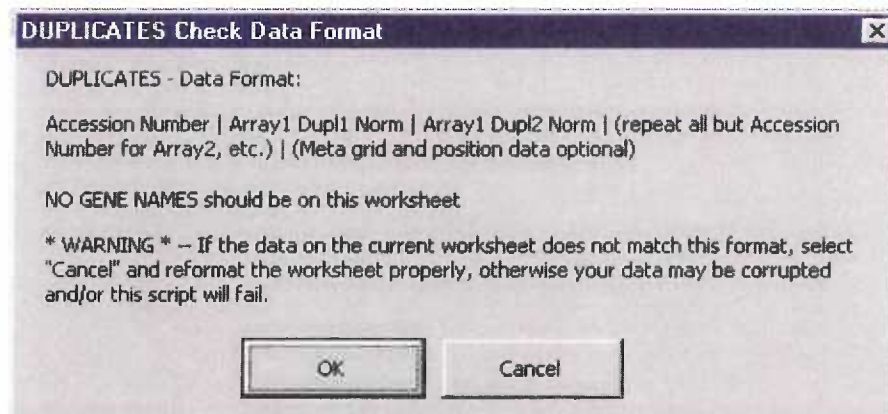
This method is currently undocumented and not recommended to be used for microarray data normalization. If you wish to use this method and have questions, contact Matthew Rodland at rodlandm@ohsu.edu or Mark Turner at turnerm@ohsu.edu

SECTION 4: ANALYZE DUPLICATES

This step in the Automated Normalization assumes that you have the normalized data worksheet in Excel created by running the **Normalize Excel Data** step as described in Section 3. If you have created and saved the spreadsheet previously, then open that spreadsheet now. The normalized data worksheet created in Section 3 should be open before running this step. Run this step only if you have duplicate arrays in your experiment and chose not to compare them manually.

Run the *Automated Normalization* program as described in [Section 0](#) with the checkbox next to *Analyze Duplicates* selected.

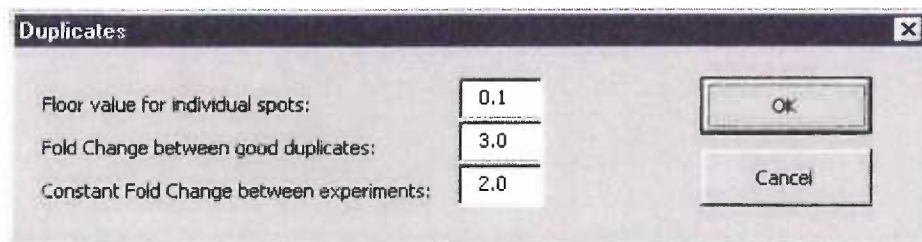
1. The **Duplicates Check Data Format** dialog box appears:



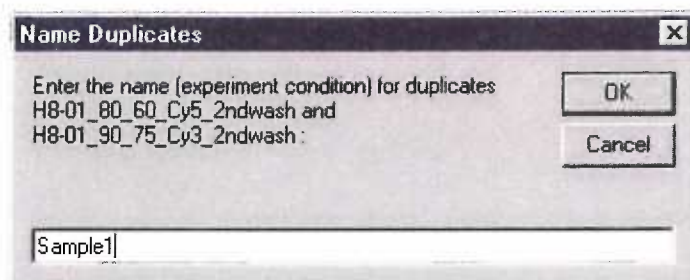
2. Make sure that the open worksheet is in the format described in the dialog box, where the vertical bars delimit each column of the worksheet, from left to right. Follow the dialog boxes instructions. The following illustration shows an example of the proper format for data prior to running the **Analyze Duplicates** step:

	A	B	C	D	E
	Accession	Array 1	Array 2	Array 3	Array 4
1	Number	Duplicate 1	Duplicate 1	Duplicate 2	Duplicate 2
2	AA123456	5.00	6.00	2.00	8.00
3					

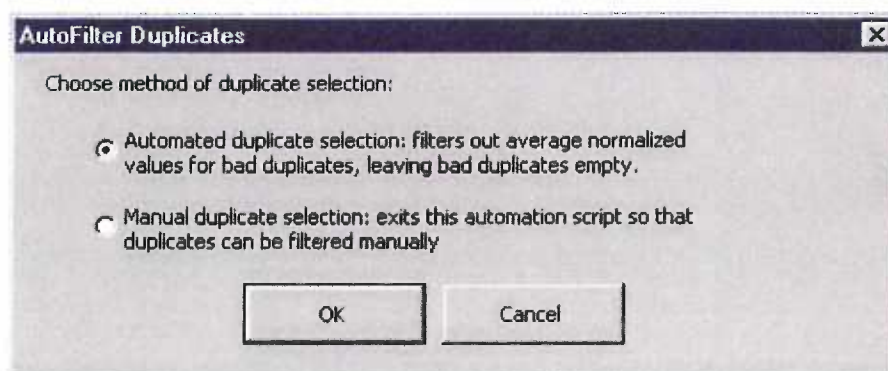
3. If the current worksheet is not in the proper format, click on **Cancel** to exit the Automated Normalization and manually edit the worksheet before running the Automated Normalization program again. Otherwise, click on **OK** to continue. The **Duplicates** dialog box appears:



4. The default settings for duplicate comparison are shown in the dialog box. You may change these to suit your particular experiment as follows:
 - a. **Floor value for individual spots:** Any accessions with normalized values below the floor value are set to the floor value before comparing duplicates. This assumes any accessions whose normalized value is below this number does not have a detectable level of expression in this experiment. The default value of 0.1 for the **20% Trimmed Mean Normalization** would set the minimum detectable level of expression to 10% of the average gene on the array.
 - b. **Fold Change between good duplicates:** This value sets what multiple is used to determine a "true" difference in expression between duplicates. The default value of 3.0 means that an accession whose normalized value is at least 3.0 times greater or less than its duplicate is considered to be differentially expressed, thus the duplicates do not match. Any duplicates that fail this comparison have their duplicate flag labeled as *False* on the worksheet, meaning that this accession has a bad duplicate.
 - c. **Constant Fold Change between experiments:** This value sets what multiple is used to determine which accessions have no measurable change in expression across the entire experiment (all arrays). The default value of 2.0 means that an accession whose maximum normalized value across all arrays is not at least 2.0 times greater than its minimum normalized value across all arrays has no measureable change in expression across the entire experiment. Any accessions that have no measureable change have their Constant FC (fold change) flag set to *False* on the worksheet, meaning that the accession will be excluded from analysis by default.
5. Click on **OK** to continue, or click **Cancel** to exit the Automated Normalization.
6. For each pair of duplicates, the **Name Duplicates** dialog box appears:



7. Type a descriptive name for the two arrays named in the dialog text so you can identify the results. It is highly recommended that you name these with a descriptive label of the experimental condition used for this array pair.
8. If the two arrays listed in the dialog text are not duplicates:
 - a. Click **Cancel** to exit the Automated Normalization.
 - b. Manually edit the normalized data worksheet so that duplicate pairs are always located in adjacent columns, starting one column to the right of the *Accession Number*.
 - c. If an array has no duplicate, either place a copy of the same array in the next column to the right, or move the column to a separate worksheet then move it back after running the **Analyze Duplicates** step of the Automated Normalization.
 - d. Run the **Analyze Duplicates** step of the Automated Normalization again and go back to step 1 above.
9. Click on **OK** to continue, or **Cancel** to exit the Automated Normalization and end here.
10. If there are more duplicates to name, you will see the **Name Duplicates** dialog box again, repeat steps 7 to 10 above.
11. After a few minutes of processing, the **AutoFilter Duplicates** dialog box appears:



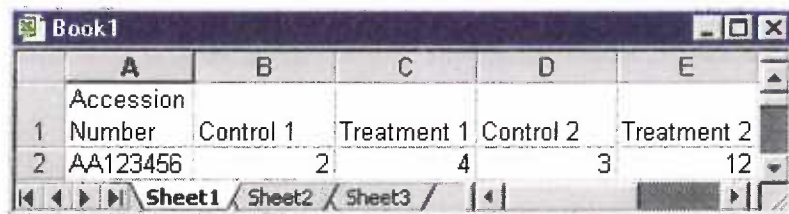
12. You have two choices here:

- a. In most cases, you will want to click **OK** with the **Automated duplicate selection** option selected. This will filter the duplicates automatically, replacing any accessions on each array whose *Dupl* (duplicate) flag or *Const_FC* (constant fold change) flag is *False* (meaning *bad*) with a null (blank) value. Go on to step 14.
 - b. Alternately, you can select the **Manual duplicate selection** option and click **OK**. This option will exit the Automated Normalization so that you can manually filter the averaged duplicates using the *Dupl* and *Const_FC* flags. If you choose this option, you have completed the **Analyze Duplicates** step of the Automated Normalization and can stop here.
13. A new worksheet named *dupl_data* is created automatically, listing the filtered accessions and average normalized values for each array.

SECTION 5: CALCULATE RATIOS (BY SLIDE)

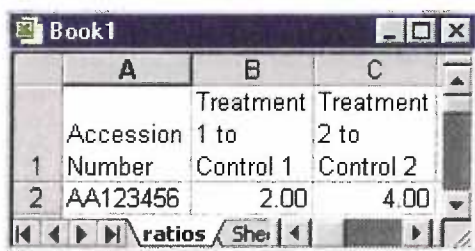
This step in the Automated Normalization assumes that you either have the normalized data worksheet in Excel created by running the **Normalize Excel Data** step as described in [Section 3](#), or analyzed duplicates as described in [Section 4](#). If you have created and saved the spreadsheet previously, then open that spreadsheet now. The normalized data worksheet created in [Section 3](#) or filtered duplicate data worksheet created in [Section 4](#) should be open before running this step.

The data on the open worksheet should be arranged (manually) so that each pair of arrays (experimental condition) you wish to compare are next to each other. Below is an example of the proper worksheet format before running this step:



	A	B	C	D	E
	Accession				
1	Number	Control 1	Treatment 1	Control 2	Treatment 2
2	AA123456	2	4	3	12

The output worksheet after running the **Calculate Ratios (by slide)** step of the Automated Normalization using the example above is as follows:



	A	B	C
	Accession	Treatment 1 to	Treatment 2 to
1	Number	Control 1	Control 2
2	AA123456	2.00	4.00

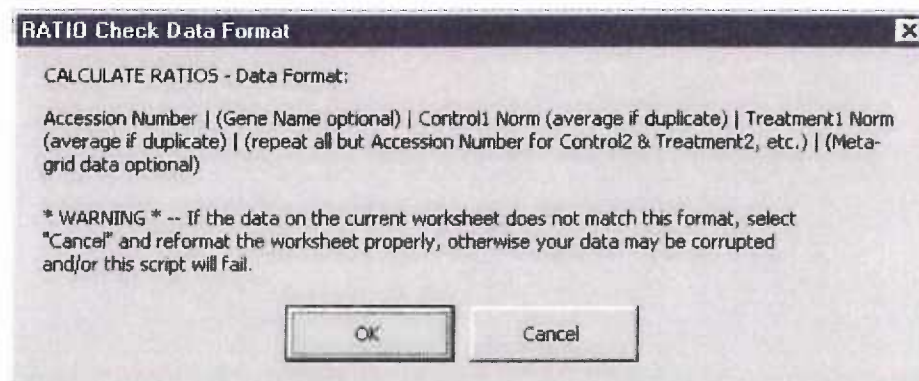
In this example, the ratios are calculated for AA123456 as:

$$\text{Treatment 1 / Control 1} = 4 / 2 = 2.00$$

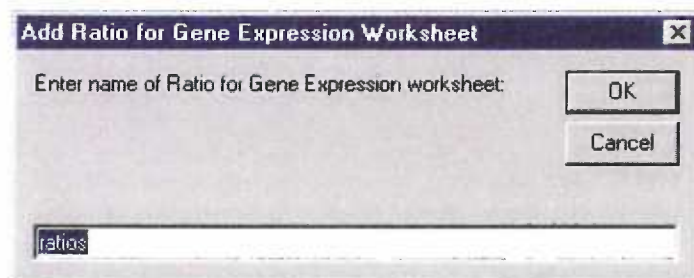
$$\text{Treatment 2 / Control 2} = 12 / 3 = 4.00$$

Run the *Automated Normalization* program as described in [Section 0](#) with the checkbox next to *Calculate Ratios (by slide)* selected.

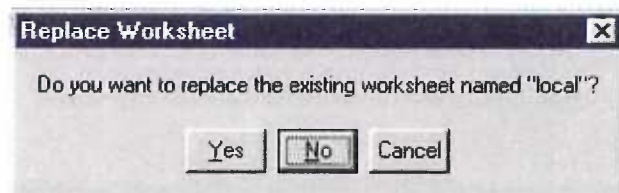
1. The **Calculate Ratios** dialog box appears:



2. Make sure that the open worksheet is in the format described in the dialog box (pictured above), where the vertical bars delimit each column of the worksheet, from left to right. Follow the dialog boxes instructions.
3. If the current worksheet is not in the proper format, click on **Cancel** to exit the Automated Normalization and manually edit the worksheet before running the Automated Normalization program again. Otherwise, click on **OK** to continue.
4. The **Add Ratio for Gene Expression Worksheet** dialog box appears:



5. Type the name of the worksheet for calculating and storing the normalized data, or accept the default name. Remember that an Excel *worksheet* is one page of an Excel *workbook*, accessible by clicking on the tab with the given name of the *worksheet* at the bottom of the screen. An Excel *workbook* is the complete Excel file as stored on disk, containing one or more *worksheets*.
6. Click on **OK** to continue, or **Cancel** to exit the Automated Normalization.
7. If a worksheet by the same name already exists, you will see the following dialog box, otherwise the **Normalize Excel Data** step is done:



8. Choose from the following:
 - a. Click **Yes** if you wish to delete the existing worksheet with the same name (shown in quotes) and replace it with the new data.
 - b. Click **No** if you wish to go back to step 4 and rename the new worksheet.
 - c. Click **Cancel** to stop the Automated Normalization and return to Excel.
9. Ratios will be calculated automatically for each pair of arrays in order from left to right. The ratio is calculated as the second column divided by the first column (right / left). Refer to the start of this section for an example.

SECTION 6: ADD GENE NAME TO ACCESSION NUMBER

This step assumes that you have a list of Accessions in the leftmost column (Column A) of the open worksheet. The program automatically matches the current gene names with each Accession, and inserts the results in the second column (Column B), shifting all other columns to the right. If this step is skipped, it is performed in the **Load Oracle for GenExplore** program after the Automated Normalization.

Warning: This step will take approximately 7 minutes to run.

Run the *Automated Normalization* program as described in [Section 0](#) with the checkbox next to *Add Gene Name to Accession Number* selected.

SECTION 7: SORT RESULTS BY ACCESSION NUMBER

This step will simply sort all the data rows in order of Accession Number.

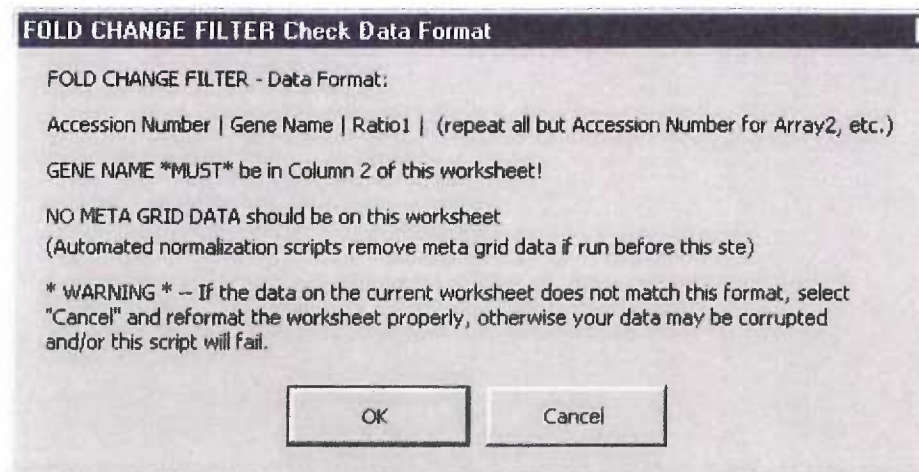
Run the *Automated Normalization* program as described in [Section 0](#) with the checkbox next to *Sort Results by Accession Number* selected.

SECTION 8: FILTER RATIOS BY FOLD CHANGE

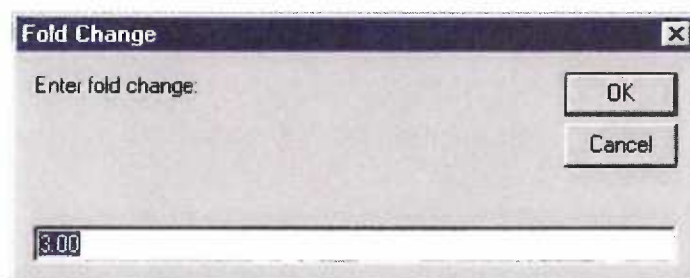
This step in the Automated Normalization assumes that you have the *Accession Numbers* in the first column (Column A), followed by the *Gene Name* in the second column (Column B) as generated in [Section 6](#), followed by a series of columns of ratio comparisons as generated in [Section 5](#). If you have created and saved the spreadsheet previously, then open that spreadsheet now. If there are columns containing Meta-grid data (*Meta_row*, *Meta_col*, *Sub_row*, *Sub_col*, or *Position*), please remove them from the open worksheet before running this step. The result is to give a list of all accessions whose ratio exceeds a given fold change in expression for each comparison in the experiment.

Run the *Automated Normalization* program as described in [Section 0](#) with the checkbox next to *Filter Ratios by Fold Change* selected.

1. The **Fold Change Filter Check Data Format** dialog box appears:



2. Make sure that the open worksheet is in the format described in the dialog box, where the vertical bars delimit each column of the worksheet, from left to right. Follow the dialog boxes instructions.
3. If the current worksheet is not in the proper format, click on **Cancel** to exit the Automated Normalization and manually edit the worksheet before running the Automated Normalization program again. Otherwise, click on **OK** to continue.
4. The **Fold Change** dialog box appears:



5. Type in the desired fold change where *normalized ratio* is determined to be differentially expressed for any accession in the experiment, or accept the default value of 3.00. Click on **OK** to continue.
6. A new worksheet named *fold_data* is created, and for each ratio comparison in the experiment, a subset of the accessions are listed in order of fold change from largest to smallest (fold change increase of expression for values > value entered in **Fold Change** dialog box, fold change decrease of expression for values < the inverse of the value entered in the **Fold Change** dialog box) only for those ratios exceeding the cutoff fold change (e.g. - greater than 3 fold change in expression).

APPENDIX D – DOCUMENTED AUTOMATED NORMALIZATION PROGRAM SOURCE CODE

```
Global dupl_form As Integer
Global rawDataName
Global normDataName
Global duplDataName
Global FormOK As Boolean

Sub Load_Oracle_For_GenExplore()
'
' Load_Oracle_For_GenExplore Macro
' Macro recorded 6/5/2000 by Mark Turner
' 10-Jan-01 M Turner check to make sure don't have duplicate column
'      headings (experiment_id's).
'
'
'
' This macro creates the files needed to load the data into the
' "GenExplore_Data" and "GenExplore_Experiments" tables in Oracle.
' These files include Oracle SQLLOADER .ctl files with the control
' statements at the top followed by the data. The macro also
' creates an SQL file to lookup the gene_names, and a DOS batch file
' to run the whole thing. All the files are called something like
' load_genexplore_....
'
' This is intended to be run when you already have the several
' experiments aligned in different columns. If the experiments
' are all in the same column, then (under certain assumptions
' as to format) the batch file load_genexplore_raw.bat may work.

Dim has_gene_name As Boolean
Dim test_range As Range
Dim array_ids(1 To 30) As String

p$ = ActiveWorkbook.Path
Range("A1").Select
' if cell B3 is not a number, then assume that column B
' contains the names of the genes
Set test_range = Range("B3")
has_gene_name = False
has_gene_name = (WorksheetFunction.IsText(test_range) Or _
```

```

WorksheetFunction.IsNA(test_range))
Open p$ + "\load_genexplore_experiments.ctl" For Output As #1
Print #1, "LOAD DATA"
Print #1, "INFILE *"
Print #1, "REPLACE"
Print #1, "INTO TABLE GenExplore_Experiments"
Print #1, "FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
Print #1, "(column_number, experiment_id)"
Print #1, "BEGINDATA"
' put experiment names in control file
If has_gene_name Then
    Selection.Offset(0, 2).Select
Else
    Selection.Offset(0, 1).Select
End If
s$ = Selection.Value
c# = 1
array_ids(c#) = s$
While (s$ <> "") And (c# < 31)
    For i = 1 To (c# - 1)
        If s$ = array_ids(i) Then
            msg$ = "Duplicate column headers found." + Chr(10)
            msg$ = msg$ + "Each column should be uniquely identified."
            MsgBox (msg$)
        End If
    Next i
    Write #1, c#, s$
    Selection.Offset(0, 1).Select
    s$ = Selection.Value
    c# = c# + 1
    array_ids(c#) = s$
Wend
cmax# = c# - 1
Close #1
'Save the Excel workbook as a text file
'n$ = ActiveWorkbook.Name
'n$ = Replace(n$, "xls", "txt")
' ActiveWorkbook.SaveAs Filename:=p$ + "\" + n$, FileFormat:=xlText
'Create file to load this text file of data into Oracle
Open p$ + "\load_genexplore_data.ctl" For Output As #1
Print #1, "OPTIONS(SKIP=1)"

```

```

Print #1, "LOAD DATA"
Print #1, "INFILE *"
Print #1, "REPLACE"
Print #1, "INTO TABLE GenExplore_Data"
Print #1, "FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'"'"
Print #1, "TRAILING NULLCOLS "
Print #1, "(accession_number,"
If has_gene_name Then
    Print #1, "gene_name , "
End If
For c# = 1 To cmax# - 1
    d$ = "data" + Trim(Str(c#))
    Print #1, d$ + " NULLIF " + d$ + "=BLANKS,"
Next c#
d$ = "data" + Trim(Str(cmax#))
Print #1, d$ + " NULLIF " + d$ + "=BLANKS"
Print #1, ")"
Print #1, "BEGINDATA"
' if gene_name in file, then data columns are one more column to the right
If has_gene_name Then
    cmin# = 2
    cmax# = cmax# + 1
Else
    cmin# = 1
End If
' write headers to control file
Range("A1").Select
out$ = Selection.Value
If has_gene_name Then
    s$ = Selection.Offset(0, 1).Value
    out$ = out$ + "," + s$
End If
For c# = cmin# To cmax#
    If WorksheetFunction.IsNumber(Selection.Offset(0, c#)) Then
        out$ = out$ + "," + Str(Selection.Offset(0, c#).Value)
    Else
        out$ = out$ + "," + Selection.Offset(0, c#).Value
    End If
Next
Print #1, out$
' write data to the control file
Selection.Offset(1, 0).Select

```

```

out$ = Selection.Value
While (out$ <> "")
    out$ = "" + out$ + ""
    If has_gene_name Then
        If WorksheetFunction.IsNonText(Selection.Offset(0, 1).Value) Then
            out$ = out$ + ", "N/A""
        Else
            s$ = Selection.Offset(0, 1).Value
            s$ = WorksheetFunction.Substitute(s$, "", "")
            out$ = out$ + ", " + s$ + ""
        End If
    End If
    For c# = cmin# To cmax#
        If WorksheetFunction.IsNumber(Selection.Offset(0, c#)) Then
            out$ = out$ + ", " + Str(Selection.Offset(0, c#).Value)
        Else
            out$ = out$ + ", "
        End If
    Next
    Print #1, out$
    Selection.Offset(1, 0).Select
    out$ = Selection.Value
Wend
Close #1
' create a SQL plus file to lookup the gene_name from accession number
Open p$ + "\load_genexplore.sql" For Output As #1
Print #1, "UPDATE Genexplore_Data gd SET (gene_name, gene_symbol,"
Print #1, "    broad_annotation_code_list, function_list,"
Print #1, "    broad_function_list, biological_process_list,"
Print #1, "    broad_biological_process_list,"; cellular_component_list, ""
Print #1, "    broad_cellular_component_list, phenotype_list,"
Print #1, "    chromosome, cytogenetic_position, summary_function,"
Print #1, "    other_annotation_list,tissues_list) = ("
Print #1, "SELECT nvl(gs.gene_name,substr(gd.gene_name,1,100)), "
Print #1, "    gene_symbol, broad_annotation_code_list, function_list,"
Print #1, "    broad_function_list, biological_process_list,"
Print #1, "    broad_biological_process_list,"; cellular_component_list, ""
Print #1, "    broad_cellular_component_list, "
Print #1, "    phenotype||decode(gs.phenotype_alternate,null,null,', ')||"
Print #1, "    phenotype_alternate,"
Print #1, "    chromosome, cytogenetic_position, summary_function,"
Print #1, "    other_annotation_list,tissues_where_expressed "

```

```

Print #1, " FROM gene_accession_numbers gan, gene_symbols gs "
Print #1, " WHERE gan.accession_number(+) = gd.accession_number||' ' "
Print #1, " AND gs.unigene_cluster_id(+) = gan.unigene_cluster_id||' ' "
Print #1, " AND rownum < 2) "
Print #1, "/"
Print #1, "COMMIT"
Print #1, "/"
Print #1, "EXIT"
Close #1

' Create a DOS batch file to run SQL LOADER to load this data
Open p$ + "\load_genexplore.bat" For Output As #1
Print #1, "sqlldr80.exe expruser/cvb48IKM@expr control=
load_genexplore_experiments.ctl"
Print #1, "sqlldr80.exe expruser/cvb48IKM@expr control= load_genexplore_data.ctl"
Print #1, "sqlplus expruser/cvb48IKM@expr @load_genexplore.sql"
Print #1, "pause"
Close #1

End Sub

Private Sub Add_Gene_Name_To_Accession_Numbers()
'
' Add_Gene_Name_To_Accession_Number Macro
' Macro updated 2/6/2001 by Matthew Rodland
' Macro recorded 4/28/2000 by Mark Turner
'
' This macro does a lookup using the accession number to find the
' gene name. The lookup is done in a separate spreadsheet.
'
Dim w As Worksheet
Dim wbactive As Workbook
Dim wb As Workbook
'
' close all worksheets except the current one
Set w = ActiveSheet
Set wbactive = ActiveWorkbook
Filename$ = ActiveWorkbook.FullName
wname$ = "Accession Numbers.xls"
' bring in "accession numbers" worksheet for the lookup
directory$ = "P:\Accession numbers\"
Workbooks.Open Filename:=directory$ + wname$

```

```

wbactive.Activate
w.Activate
' put in formula to do the lookup
Columns("B:B").Select
Selection.Insert Shift:=xlToRight
Range("B1").Select
ActiveCell.FormulaR1C1 = _
    "=VLOOKUP(RC[-1],'" + wname$ + "'!Query_from_expr_mandrake,2,FALSE)"
Selection.Offset(1, -1).Select
Selection.End(xlDown).Select
Selection.Offset(0, 1).Select
Range("A1", Selection).Select
Selection.Table ColumnInput:=Range("A1")
' turn off automatic recalculation
With Application
    .Calculation = xlManual
    .MaxChange = 0.001
End With
ActiveWorkbook.PrecisionAsDisplayed = False
'
'   Change gene names to constants.
Columns("C:C").Select
Selection.Insert Shift:=xlToRight
Columns("B:B").Select
Selection.Copy
Columns("C:C").Select
Selection.PasteSpecial Paste:=xlValues, Operation:=xlNone, SkipBlanks:= _
    False, Transpose:=False
Range("C1").Value = "Gene Name"
Columns("B:B").Select
Application.CutCopyMode = False
Selection.Delete Shift:=xlToLeft
' turn auto recalculation back on
With Application
    .Calculation = xlAutomatic
    .MaxChange = 0.001
End With
ActiveWorkbook.PrecisionAsDisplayed = False
'
'   Name the range "expression_database"
Range("A1").Select
ActiveWorkbook.Names.Add Name:="expression_database", RefersToR1C1:= _

```

```

        Selection.CurrentRegion
Range("A1").Select
' close the accession lookup database
Workbooks(wname$).Close savechanges:=False
Filename$ = WorksheetFunction.Substitute(Filename$, ".txt", ".xls")
ActiveWorkbook.SaveAs Filename:=Filename$, FileFormat:= _
    xlNormal, Password:="", WriteResPassword:="", ReadOnlyRecommended:=False _
    , CreateBackup:=False

End Sub

Private Sub Raw_data_to_Excel()
'
' Raw_data_to_Excel Macro
' Macro recorded 9/12/2000 by Matthew J. Rodland
' Last updated 7/17/2001 by Matthew J. Rodland

' This macro reads in the raw data file as exported from the Barry Microarray
' Data Warehouse (*.txt file), formats and saves the file for analysis
' in Excel.

Dim todays_date As Date

Rows("1:1").Select
With Selection
    .HorizontalAlignment = xlGeneral
    .VerticalAlignment = xlBottom
    .WrapText = True
    .Orientation = 0
    .AddIndent = False
    .ShrinkToFit = False
    .MergeCells = False
End With
Columns("A:A").Select
Selection.ColumnWidth = 9
Columns("B:B").Select
Range(Selection, Selection.End(xlToRight)).Select
Selection.NumberFormat = "0.00"

Range("A2").Select
ActiveWindow.FreezePanes = True

```

```

rawData = "norm_data " + Replace(CStr(Date), "/", "-") + ".xls"
rawData = Application.GetSaveAsFilename(rawData, _
    "Microsoft Excel Workbook (*.xls), *.xls", , _
    "Save Data As:")
If rawData <> False Then
    ActiveWorkbook.SaveAs Filename:=rawData, _
        FileFormat:=xlNormal, Password:="", _
        WriteResPassword:="", ReadOnlyRecommended:= _
        False, CreateBackup:=False
    rawDataName = ActiveWindow.Caption
End If
End Sub

Private Sub Slidewise_Trimmed_Norm()

' Slidewise_Trim_Norm macro
' Macro written by Matthew J. Rodland
' Last updated 7/17/2001 by Matthew J. Rodland

' This macro performs a 20% slidewise trimmed mean normalization on an Excel
' worksheet containing Local corrected data. Saves the result in a new worksheet
' in the same Excel workbook.

    Dim local_sheet As Worksheet
    Dim local_range As Range
    Dim norm_sheet As Worksheet
    Dim myName As String
    Dim numRows As Integer
    Dim numCols As Integer

' Local_corrected
Set local_sheet = ActiveSheet
myName = newWorksheetName("20% Trimmed Mean Normalization", "trim20_norm")
Set norm_sheet = Sheets.Add
norm_sheet.Name = myName

numRows = local_sheet.Range("A1").End(xlDown).row
numCols = local_sheet.Range("A1").End(xlToRight).Column
local_sheet.Range("1:1").Copy
norm_sheet.PasteSpecial
For colIter = 1 To numCols
    If Right(Trim(local_sheet.Cells(1, colIter).Value), 9) = "corrected" Then

```

```

norm_sheet.Cells(1, colIter).Value = _
    Replace(norm_sheet.Cells(1, colIter).Value, _
        "Local corrected", "Slidewise Trimmed Norm")
Set local_range = local_sheet.Cells(2, colIter)
Set normFactor = local_range.Cells(numRows).Offset(2, 0)
normFactor.Formula = _
    "=trimmean(" & local_range.Address(False, False) _
    & ":" & local_range.Cells(numRows - 1).Address(False, False) _
    & ",0.2)"

For rowIter = 2 To numRows
    If local_sheet.Cells(rowIter, colIter).Value <> "" Then
        norm_sheet.Cells(rowIter, colIter).Value = _
            local_sheet.Cells(rowIter, colIter).Value / _
            normFactor.Value
    End If
Next rowIter
Else
    local_sheet.Cells(1, colIter).EntireColumn.Copy
    norm_sheet.Cells(1, colIter).PasteSpecial
End If
Next colIter
norm_sheet.Cells.NumberFormat = "0.00"
norm_sheet.Range("A1").Select
End Sub

Private Sub Process_dupl_slides()
'
' Process_dupl_slides Macro
' Macro recorded 9/18/2000 by Matthew J. Rodland
' Last updated 7/17/2001 by Matthew J. Rodland

' This macro reads the currently open worksheet formatted for duplicate comparison.
' Duplicate pairs are averaged and tested for Duplicate Fold Change, Constant Fold
' Change, and Floor values entered by the user. Those tests are stored in their own
' boolean flag columns on the new worksheet added to the current workbook.

' Input worksheet format:
' column 1 - Accession number
' column 2 & 3, 4 & 5, etc. - Each pair of duplicate slides
' *NOTE: ALL SLIDES MUST HAVE A DUPLICATE FOR THIS VERSION*

```

```

Dim FloorVar
Dim ConstFCVar
Dim DuplFCVar

Dim dupl_1st_name As String
Dim dupl_2nd_name As String
Dim ID_Name As String
Dim num_rows As Integer

Load Duplicates
Duplicates.Show
If FormOK = False Then End
FloorVar = Duplicates.Floor.Value
DuplFCVar = Duplicates.FC_Dupl.Value
ConstFCVar = Duplicates.FC_Const.Value

Set norm_sheet = ActiveSheet
duplDataName = newWorksheetName("Duplicates", "dupl_compare")
Set dupl_sheet = Sheets.Add
dupl_sheet.Name = duplDataName

norm_sheet.Cells.Copy
dupl_sheet.Cells.PasteSpecial Paste:=xlValues, Operation:=xlNone, SkipBlanks:=
-
    False, Transpose:=False
Rows("1:1").Select
Application.CutCopyMode = False
With Selection
    .HorizontalAlignment = xlGeneral
    .VerticalAlignment = xlBottom
    .WrapText = True
    .Orientation = 0
    .AddIndent = False
    .ShrinkToFit = False
    .MergeCells = False
End With
Columns("A:A").ColumnWidth = 9
Cells.Select
Selection.NumberFormat = "0.00"

num_rows = Range("A1").End(xlDown).row

```

```

Count# = 0
Range("A1").Select
Selection.Offset(0, 2).Select
While Selection.Value <> "" And Left(Trim(Selection.Value), 4) <> "meta" _
And Left(Trim(Selection.Value), 3) <> "sub" _
And Left(Trim(Selection.Value), 8) <> "position"
    Count# = Count# + 1
    Selection.Offset(0, 2).Select
Wend
Range("D1").Select
Selection.EntireColumn.Insert
Selection.EntireColumn.Insert
dupl_1st_name = Split(ActiveCell.Offset(0, -2).Value, " - ")(0)
dupl_2nd_name = Split(ActiveCell.Offset(0, -1).Value, " - ")(0)
ID_Name = InputBox("Enter the name (experiment condition) for duplicates " & _
    dupl_1st_name & " and " & dupl_2nd_name & " : ", _
    "Name Duplicates", "1")
ActiveCell.FormulaR1C1 = "Dupl " & ID_Name & "?"
ActiveCell.Offset(0, 1).FormulaR1C1 = "Average " & ID_Name

For X# = 2 To Count#
    Selection.Offset(0, 4).Select
    Selection.EntireColumn.Insert
    Selection.EntireColumn.Insert
    dupl_1st_name = Split(ActiveCell.Offset(0, -2).Value, " - ")(0)
    dupl_2nd_name = Split(ActiveCell.Offset(0, -1).Value, " - ")(0)
    ID_Name = InputBox("Enter the name (experiment condition) for duplicates "
& _
        dupl_1st_name & " and " & dupl_2nd_name & " : ", _
        "Name Duplicates", Str(X#))
    ActiveCell.FormulaR1C1 = "Dupl " & ID_Name & "?"
    ActiveCell.Offset(0, 1).FormulaR1C1 = "Average " & ID_Name
    With ActiveCell.Characters(Start:=1, Length:=6).Font
        .Name = "Arial"
        .FontStyle = "Regular"
        .Size = 10
        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
    End With

```

```

        .ColorIndex = xlAutomatic
    End With
Next X#

Range("A1").Select
Selection.End(xlDown).Select
total_row# = Selection.row
Selection.Offset(2, 0).Select
ActiveCell.FormulaR1C1 = "Floor"
Selection.Offset(1, 0).Select
ActiveCell.FormulaR1C1 = "Dupl FC"
Selection.Offset(1, 0).Select
ActiveCell.FormulaR1C1 = "Const FC"
Selection.Offset(-2, 1).Select
ActiveWorkbook.Names.Add Name:="Floor", RefersToR1C1:=Selection
ActiveCell.FormulaR1C1 = FloorVar
Selection.Offset(1, 0).Select
ActiveWorkbook.Names.Add Name:="Dupl_FC", RefersToR1C1:=Selection
ActiveCell.FormulaR1C1 = DuplFCVar
Selection.Offset(1, 0).Select
ActiveWorkbook.Names.Add Name:="Const_FC", RefersToR1C1:=Selection
ActiveCell.FormulaR1C1 = ConstFCVar
Range("D2").Select
ActiveCell.FormulaR1C1 = _
    "=IF(OR(RC[-2]="""",RC[-1]=""""),False,IF(MAX(RC[-2],Floor,RC[-1])/MIN(MAX(RC[-2],Floor),MAX(Floor,RC[-1]))>Dupl_FC,False,True))"
Range("D2").Select
Selection.Copy
Range(Selection, Selection.Offset(total_row# - 2)).Select
ActiveSheet.Paste
ActiveCell.Offset(total_row#).FormulaR1C1 = "=COUNTIF(R2C:R[-2]C,""&FALSE&"" )"
For X# = 2 To Count#
    ActiveCell.Offset(0, 4).Select
    Range(Selection, Selection.Offset(total_row# - 2)).Select
    ActiveSheet.Paste
    ActiveCell.Offset(total_row#).FormulaR1C1 = "=COUNTIF(R2C:R[-2]C,""&FALSE&"" )"
Next X#

Range("E2").Select
ActiveCell.FormulaR1C1 = "=AVERAGE(MAX(RC[-3],Floor),MAX(RC[-2],Floor))"
Selection.Copy

```

```

Selection.Offset(0, -1).Select
Selection.End(xlDown).Select
Selection.Offset(0, 1).Select
Range(Selection, Selection.End(xlUp)).Select
ActiveSheet.Paste
Selection.End(xlUp).Select
Range(Cells(2, 5), Cells(num_rows, 5)).Select
Application.CutCopyMode = False
Selection.Copy
For X# = 2 To Count#
    Selection.Offset(0, 4).Select
    ActiveSheet.Paste
Next X#

Selection.Offset(-1, 1).Select
Selection.EntireColumn.Insert
ActiveCell.FormulaR1C1 = "Const FC?"
Selection.Offset(1, 0).Select
myFormula$ = "RC[-1]"
For X# = 2 To Count#
    myFormula$ = myFormula$ + ",RC[-" + LTrim(Str((X# - 1) * 4) + 1)) + "]"
Next X#
ActiveCell.FormulaR1C1 = _
    "=IF(MAX(" + myFormula$ + ")/MIN(" + myFormula$ + "<Const_FC,False,True)"
ActiveCell.Copy
Range(Cells(2, Selection.Column), Cells(num_rows,
Selection.Column)).PasteSpecial
Cells(num_rows, Selection.Column).Select
Selection.Offset(2, 0).Select
Application.CutCopyMode = False
ActiveCell.FormulaR1C1 = "=COUNTIF(R2C:R[-2]C,""&FALSE&"")"
Range("A1").Select

ActiveWorkbook.Save
End Sub

Sub Automated_Normalization()

' Automated_Normalization Macro
' Macro written by Matthew J. Rodland
' Last updated 7/17/2001 by Matthew J. Rodland

```

```
' This macro is the main Automated Normalization program, and as such is the only
' normalization related macro visible to the end user on the Macros list.
' This program displays the main GUI to the user, and runs each step of the
' Normalization selected by the user.
```

```
Load Analysis
Analysis.Show
If FormOK <> True Then
    Exit Sub
Else
    If Analysis.ImportText Then
        Import_text
        Raw_data_to_Excel
    End If
    If Analysis.LocalCorrected Then Local_corrected
    If Analysis.Normalize Then
        If Analysis.Trimmean10 Then
            Load Format_Trimmean
            Format_Trimmean.Show
            If FormOK = False Then End
            Slidewise_Trimmed_Norm
        End If
        If Analysis.Gridwise Then
            Load Format_Gridwise
            Format_Gridwise.Show
            If FormOK = False Then End
            Norm_by_MetaGrid
        End If
    End If
    If Analysis.Duplicates Then
        If Analysis.TwoSlide Then
            Load Format_Dupl
            Format_Dupl.Show
            If FormOK = False Then End
            Process_dupl_slides
            Load AutoFilter
            AutoFilter.Show
            If AutoFilter.ManualDuplFilter = True _
            Or FormOK = False Then End
            If AutoFilter.AutoDuplFilter = True Then Filter_Dupl
        End If
    End If
End If
```

```

        If Analysis.CalcRatio Then
            Load Format_Ratio
            Format_Ratio.Show
            If FormOK = False Then End
            do_ratio_2
        End If

        If Analysis.AddName Then Add_Gene_Name_To_Accession_Numbers
        If Analysis.SortAccNum Then Sort_by_Accession
        If Analysis.FoldChange Then
            Load Format_FC
            Format_FC.Show
            If FormOK = False Then End
            fold_change
        End If
        If Analysis.Dupl_Graph Then Graph_Duplicates
        If Analysis.Quality_Summary Then Quality_Summary
    End If
End Sub

Private Sub Local_corrected()

    ' Local_corrected Macro
    ' Macro written by Matthew J. Rodland
    ' Last updated 7/17/2001 by Matthew J. Rodland

    ' This macro reads the currently active raw data worksheet in Excel, calculates and
    ' returns the data locally corrected for background on a new worksheet in the
    ' same workbook.

    Dim raw_sheet As Worksheet
    Dim raw_range As Range

    Dim local_sheet As Worksheet
    Dim local_range As Range

    Dim meta_row_loc As Range
    Dim meta_col_loc As Range
    Dim sub_row_loc As Range
    Dim sub_col_loc As Range

    Dim signal_value

```

```

Dim background_value
Dim selected_flag

Dim acc_row As Integer
Dim acc_count As Integer

Dim myName As String
Dim pos_flag As Boolean
Dim meta_flag As Integer

Set raw_sheet = ActiveSheet
Application.CutCopyMode = False

myName = newWorksheetName("Local corrected", "local")

'copy the column of accession numbers
raw_sheet.Columns("A:A").Copy
' create a new sheet and name it "local"
Set local_sheet = Sheets.Add
local_sheet.Name = myName
' paste in the column of accession numbers
local_sheet.Range("A1").Select
local_sheet.Paste
' count how many rows there are to calculate
acc_count = Range("A1", Range("A1").End(xlDown)).Rows.Count
' format the first row of the local sheet
With local_sheet.Rows("1:1")
    .HorizontalAlignment = xlGeneral
    .VerticalAlignment = xlBottom
    .WrapText = True
    .Orientation = 0
    .AddIndent = False
    .ShrinkToFit = False
    .MergeCells = False
End With
' start in column B on both sheets
Set raw_range = raw_sheet.Range("B1")
Set local_range = local_sheet.Range("B1")
'

'test for presence of "position" column
pos_flag = False

```

```

While raw_range.Value <> ""
    If Trim(Right(raw_range.Value, 8)) = "position" Then
        raw_range.Value = "position"
        raw_range.EntireColumn.Copy
        local_sheet.Range("A1").EntireColumn.Insert
        raw_range.EntireColumn.Delete Shift:=xlToLeft
        Set raw_range = raw_sheet.Range("A1").End(xlToRight)
        pos_flag = True
    End If
    Set raw_range = raw_range.Offset(0, 1)
Wend
If pos_flag = True Then
    Set local_range = local_sheet.Range("C1")
Else
    Set local_range = local_sheet.Range("B1")
End If
Set raw_range = raw_sheet.Range("B1")

meta_flag = 0
While raw_range.Value <> ""
    substring = Trim(Right(raw_range.Value, 7))
    Select Case substring
        Case "eta_row"
            meta_flag = meta_flag + 1
            raw_range.Value = "meta_row"
            raw_range.EntireColumn.Copy
            local_range.PasteSpecial (xlPasteAll)
            Set meta_row_loc = local_range
            Set raw_range = raw_sheet.Cells(1, raw_range.Column + 1)
            Set local_range = local_sheet.Cells(1, local_range.Column + 1)
        Case "eta_col"
            meta_flag = meta_flag + 1
            raw_range.Value = "meta_col"
            raw_range.EntireColumn.Copy
            local_range.PasteSpecial (xlPasteAll)
            Set meta_col_loc = local_range
            Set raw_range = raw_sheet.Cells(1, raw_range.Column + 1)
            Set local_range = local_sheet.Cells(1, local_range.Column + 1)
        Case "sub_row"
            meta_flag = meta_flag + 1
            raw_range.Value = "sub_row"
            raw_range.EntireColumn.Copy

```

```

        local_range.PasteSpecial (xlPasteAll)
        Set sub_row_loc = local_range
        Set raw_range = raw_sheet.Cells(1, raw_range.Column + 1)
        Set local_range = local_sheet.Cells(1, local_range.Column + 1)
    Case "sub_col"
        meta_flag = meta_flag + 1
        raw_range.Value = "sub_col"
        raw_range.EntireColumn.Copy
        local_range.PasteSpecial (xlPasteAll)
        Set sub_col_loc = local_range
        Set raw_range = raw_sheet.Cells(1, raw_range.Column + 1)
        Set local_range = local_sheet.Cells(1, local_range.Column + 1)
    Case Else
        ' copy the header into the first row
        acc_row = 1
        local_range.Value = Replace(raw_range.Offset(0, 2).Value, _
            "selected", "Local corrected")
        ' format the numbers in this column
        local_range.EntireColumn.NumberFormat = "0.00"
        For acc_row = 2 To acc_count Step 1
            Set raw_range = raw_range.Offset(1, 0)
            Set local_range = local_range.Offset(1, 0)
            signal_value = raw_range.Value
            background_value = raw_range.Offset(0, 1).Value
            selected_flag = raw_range.Offset(0, 2).Value
            If selected_flag = 1 Then _
                local_range.Value = signal_value - background_value
        Next acc_row
        ' move over to the next set of columns
        Set raw_range = raw_sheet.Cells(1, raw_range.Column + 3)
        Set local_range = local_sheet.Cells(1, local_range.Column + 1)
    End Select
Wend

If pos_flag <> True And meta_flag = 4 Then
    local_range.Value = "position"
    ' format the numbers in this column
    local_range.EntireColumn.NumberFormat = "0"
    For acc_row = 2 To acc_count Step 1
        Set local_range = local_range.Offset(1, 0)
        Set meta_row_loc = meta_row_loc.Offset(1, 0)
        Set meta_col_loc = meta_col_loc.Offset(1, 0)
        Set sub_row_loc = sub_row_loc.Offset(1, 0)
    
```

```

        Set sub_col_loc = sub_col_loc.Offset(1, 0)
        local_range = CStr(meta_row_loc) _
            + Format(CStr(meta_col_loc), "000") _
            + Format(CStr(sub_row_loc), "000") _
            + Format(CStr(sub_col_loc), "000")
    Next acc_row
Else
    If pos_flag = True Then
        local_sheet.Range("A:A").Copy
        local_range.EntireColumn.PasteSpecial (xlPasteAll)
        local_sheet.Range("A1").EntireColumn.Delete
    End If
End If
Cells.Select
Selection.Sort Key1:=Range("A2").End(xlToRight), Order1:=xlAscending,
Header:=xlYes, _
    OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom
local_sheet.Range("A1").Select
End Sub

Private Sub Norm_by_MetaGrid()

' Norm_by_MetaGrid Macro
' Macro written by Matthew J. Rodland
' Last updated 7/17/2001 by Matthew J. Rodland

' This macro, currently unused, reads in data from the active Local corrected
' worksheet, computes and returns the data normalized by meta-grid on a new
' worksheet in the same workbook.

Dim num_meta_rows As Integer
Dim num_meta_cols As Integer
Dim num_sub_rows As Integer
Dim num_sub_cols As Integer

Dim loop_meta_row As Integer
Dim loop_meta_col As Integer
Dim loop_sub_row As Integer
Dim loop_sub_col As Integer

Dim norm_factor

```

```

Dim local_sheet As Worksheet
Dim local_range As Range
Dim norm_sheet As Worksheet
Dim norm_range As Range
Dim meta_range As Range

Set local_sheet = ActiveSheet
Application.CutCopyMode = False
'copy the column of accession numbers
local_sheet.Columns("A:A").Copy
' create a new sheet and name it "norm"
Set norm_sheet = Sheets.Add
norm_sheet.Name = "norm"
' paste in the column of accession numbers
norm_sheet.Range("A1").Select
norm_sheet.Paste
' count how many rows there are to calculate
acc_count = Range("A1", Range("A1").End(xlDown)).Rows.Count
' format the first row of the norm sheet
With norm_sheet.Rows("1:1")
    .HorizontalAlignment = xlGeneral
    .VerticalAlignment = xlBottom
    .WrapText = True
    .Orientation = 0
    .AddIndent = False
    .ShrinkToFit = False
    .MergeCells = False
End With
Set local_range = local_sheet.Range("B1")
Set norm_range = norm_sheet.Range("B1")

num_meta_rows = 0
num_meta_cols = 0
num_sub_rows = 0
num_sub_cols = 0

local_sheet.Activate
Range("A1").Select
While Selection.Value <> ""
    Select Case Right(Selection.Value, 7)
        Case "eta_row"
            Selection.Value = "meta_row"
    End Select

```

```

        num_meta_rows = Application.WorksheetFunction.Max _
            (Selection.EntireColumn)
    Case "eta_col"
        Selection.Value = "meta_col"
        num_meta_cols = Application.WorksheetFunction.Max _
            (Selection.EntireColumn)
    Case "sub_row"
        Selection.Value = "sub_row"
        num_sub_rows = Application.WorksheetFunction.Max _
            (Selection.EntireColumn)
    Case "sub_col"
        Selection.Value = "sub_col"
        num_sub_cols = Application.WorksheetFunction.Max _
            (Selection.EntireColumn)
    End Select
    Selection.Offset(0, 1).Select
Wend

If num_meta_rows = 0 Or num_meta_cols = 0 Or _
    num_sub_rows = 0 Or num_sub_cols = 0 Then
    MsgBox ("Error: Could not locate grid information on spreadsheet")
    End
End If
norm_sheet.Activate
'

While local_range.Value <> "" _
And local_range.Value <> "meta_row" _
And local_range.Value <> "meta_col" _
And local_range.Value <> "sub_row" _
And local_range.Value <> "sub_col"
    norm_range.Value = _
        Replace(local_range.Value, "Local corrected", "norm")
    ' format the numbers in this column
    norm_range.EntireColumn.NumberFormat = "0.00"
    Set local_range = local_range.Offset(1, 0)
    Set norm_range = norm_range.Offset(1, 0)

    For loop_meta_row = 1 To num_meta_rows
        For loop_meta_col = 1 To num_meta_cols
            Set meta_range = Range(local_range,
local_range.Offset((num_sub_rows * num_sub_cols) - 1, 0))

            flag = False

```

```

    For Each element In meta_range
        If element <> "" Then
            flag = True
            Exit For
        End If
    Next element
    If flag = True Then
        norm_factor = _
        Application.WorksheetFunction.Average(meta_range)
        For loop_sub_row = 1 To num_sub_rows
            For loop_sub_col = 1 To num_sub_cols
                norm_range.Value = local_range.Value / norm_factor
                Set norm_range = norm_range.Offset(1, 0)
                Set local_range = local_range.Offset(1, 0)
            Next loop_sub_col
        Next loop_sub_row
    Else
        Set norm_range = norm_range.Offset(num_sub_rows * num_sub_cols,
0)
        Set local_range = local_range.Offset(num_sub_rows *
num_sub_cols, 0)
    End If

    Next loop_meta_col
Next loop_meta_row

' move over to the next set of columns
Set local_range = local_sheet.Cells(1, local_range.Column + 1)
Set norm_range = norm_sheet.Cells(1, norm_range.Column + 1)
Wend
Range("A1").Select

End Sub

Private Sub do_ratio_2()

' do_ratio_2 Macro
' Macro modified by Matthew J. Rodland from do_ratio_1 by Mark Turner 2/6/2001
'
' Calculate the ratio of normalized values of gene expression data
' The spreadsheet must be in the following format:
' column 1 Accession_numbers

```

```

' column 2 control value for condition 1
' column 3 treated value for condition 1
' columns 4, 5 are same as 2, 3 but for condition 2
' and so on
' The first row is taken to be the header row
' The resulting ratios are put in a separate sheet labelled "ratio_treated_to_ctrl"
' If a norm value is less than floor_value (set arbitrarily to 0.1),
' then it is set to the floor_value. This prevents the worst of the errors caused
' by noise.
' If both treated and control are less than the floor_value, then the ratio is
blank.
'

Dim norm_sheet As Worksheet
Dim norm_range As Range

Dim treated_value
Dim control_value

Dim ratio_sheet As Worksheet
Dim ratio_range As Range

Dim acc_row As Integer
Dim acc_count As Integer
Dim ratio_name As String
Dim temp As String
Dim floor_value
floor_value = 0.1 ' default value for floor_value

Set norm_sheet = ActiveSheet
Application.CutCopyMode = False
'copy the column of accession numbers
norm_sheet.Columns("A:A").Copy
' create a new sheet and name it
ratio_name = newWorksheetName("Ratio for Gene Expression", "ratios")
Set ratio_sheet = Sheets.Add
ratio_sheet.Name = ratio_name
' paste in the column of accession numbers
ratio_sheet.Range("A1").Select
ratio_sheet.Paste
' count how many rows there are to calculate
acc_count = Range("A1", Range("A1").End(xlDown)).Rows.Count
' format the first row of the ratio_treated_to_ctrl sheet

```

```

With ratio_sheet.Rows("1:1")
    .HorizontalAlignment = xlGeneral
    .VerticalAlignment = xlBottom
    .WrapText = True
    .Orientation = 0
    .AddIndent = False
    .ShrinkToFit = False
    .MergeCells = False
End With

' start in data column on both sheets, skip Gene Name if present
If norm_sheet.Range("B1").Value = "Gene Name" Then
    norm_sheet.Range("B:B").Copy
    ratio_sheet.Range("B1").PasteSpecial
    Set norm_range = norm_sheet.Range("C1")
    Set ratio_range = ratio_sheet.Range("C1")
Else
    Set norm_range = norm_sheet.Range("B1")
    Set ratio_range = ratio_sheet.Range("B1")
End If

'
While norm_range.Value <> "" _
    ' copy the header into the first row
    acc_row = 1
    temp = Left(Trim(norm_range.Value), 4)
    If temp = "meta" Or temp = "sub_" Or temp = "posi" Then
        norm_range.EntireColumn.Copy
        ratio_range.PasteSpecial
        Set norm_range = norm_range.Offset(0, 1)
        Set ratio_range = ratio_range.Offset(0, 1)
    Else
        ratio_range.Value = norm_range.Offset(0, 1).Value _
            + " to " + norm_range.Value
        ' format the numbers in this column
        ratio_range.EntireColumn.NumberFormat = "0.00"
        ' for each remaining row, calculate the ratio
        For acc_row = 2 To acc_count Step 1
            Set norm_range = norm_range.Offset(1, 0)
            Set ratio_range = ratio_range.Offset(1, 0)
            ' Make sure data is okay
            control_value = norm_range.Value
            treated_value = norm_range.Offset(0, 1).Value
            If control_value < floor_value _

```

```

        And control_value <> "" Then
            control_value = floor_value
        End If
        If treated_value < floor_value _
        And treated_value <> "" Then
            treated_value = floor_value
        End If
        If treated_value >= floor_value _
        And control_value >= floor_value _
        And treated_value <> "" _
        And control_value <> "" Then
            ' calculate the ratio
            ratio_range.Value = treated_value / control_value
        End If
    Next acc_row
    ' move over to the next set of columns
    Set norm_range = norm_sheet.Cells(1, norm_range.Column + 2)
    Set ratio_range = ratio_sheet.Cells(1, ratio_range.Column + 1)
End If
Wend

End Sub

Private Sub fold_change()

    ' Macro created 2/6/2001 by Matthew J. Rodland
    ' Last updated 7/17/2001 by Matthew J. Rodland
    ' Auto filters a given fold change for ratios by column

    ' Worksheet format as follows
    ' Column 1 : Accession Numbers
    ' Column 2 : Gene Names
    ' Column 3+ : Each column a norm ratio to filter

    Dim ratio_sheet As Worksheet
    Dim fold_sheet As Worksheet
    Dim ratio_range As Range
    Dim fold_range As Range
    Dim fold
    Dim fold_inv
    Dim row As Integer
    Dim row_count As Integer

```

```

Set ratio_sheet = ActiveSheet
' create a new sheet and name it "ratio_treated_to_ctrl"
Set fold_sheet = Sheets.Add
fold_sheet.Name = "fold_data"

Set ratio_range = ratio_sheet.Range("C1")
Set fold_range = fold_sheet.Range("A1")
row_count = ratio_sheet.Range("A1").End(xlDown).row

'Input fold change to compute, allowinging decimals
fold = CDec(InputBox("Enter fold change:", "Fold Change", "3.00"))
fold_inv = 1 / fold

'Loop for each column of ratio data
While ratio_range.Value <> ""
    'Column title
    fold_range = "Accession Number"
    fold_range.Offset(0, 1) = "Gene Name"
    fold_range.Offset(0, 2) = Format(fold, "##0.00") + " fold " + ratio_range.Value
    'Move to 1st row of data
    Set fold_range = fold_range.Offset(1, 0)
    Set ratio_range = ratio_range.Offset(1, 0)
    'Generate list of data matching fold change criteria
    For row = 2 To row_count
        If ratio_range.Value > fold _
        Or ratio_range.Value < fold_inv Then
            'Accession #
            fold_range = ratio_sheet.Cells(ratio_range.row, 1)
            'Gene Name
            fold_range.Offset(0, 1) = ratio_sheet.Cells(ratio_range.row, 2)
            'ratio matching fold change criteria
            fold_range.Offset(0, 2) = ratio_range
            'increment row of results
            Set fold_range = fold_range.Offset(1, 0)
        End If
        Set ratio_range = ratio_range.Offset(1, 0)
    Next row

    'Sort by fold change ratio
    Range(fold_sheet.Cells(1, fold_range.Column), _
        fold_sheet.Cells(fold_range.row, fold_range.Column + 2)).Select

```

```

        Selection.Sort Key1:=fold_sheet.Cells(2, fold_range.Column + 2), _
            Order1:=xlDescending, Header:=xlYes, _
            OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom

        'reset ranges for next column of ratio data
        Set fold_range = fold_sheet.Cells(1, fold_range.Column + 4)
        Set ratio_range = ratio_sheet.Cells(1, ratio_range.Column + 1)
    Wend

    Rows("1:1").WrapText = True
    fold_sheet.Range("A1").Select
End Sub

Private Sub Import_text()

    ' Import_text Macro
    ' Macro recorded 3/9/2001 by Matthew J. Rodland
    ' Last updated 7/17/2001 by Matthew J. Rodland

    ' Presents user with Open dialog box to import text file from disk.

    Dim raw_import
    raw_import = Application.GetOpenFilename("Text Files (*.txt), *.txt")

    Workbooks.OpenText Filename:= _
        raw_import, _
        Origin:=xlWindows, StartRow:=1, DataType:=xlDelimited, TextQualifier:= _
        xlDoubleQuote, ConsecutiveDelimiter:=False, Tab:=True, Semicolon:=False, _
        Comma:=False, Space:=False, Other:=False, FieldInfo:=Array(1, 1)
End Sub

Private Sub Sort_by_Accession()

    ' Sort_by_Accession Macro
    ' Macro written by Matthew J. Rodland
    ' Last updated 7/17/2001 by Matthew J. Rodland

    ' Simple script to automatically sort microarray data in a worksheet by Accession.

    Cells.Select
    Selection.Sort Key1:=Range("A2"), Order1:=xlAscending, Header:=xlYes, _
        OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom
End Sub

```

```

Private Sub raw_Near_Neighbor()

' raw_Near_Neighbor Macro
' Macro written by Matthew J. Rodland
' Last updated 7/17/2001 by Matthew J. Rodland

' Currently unused script computing a variation on the local corrected background.
' This method combines the background of nearby spots in a 3x3 grid around
' the signal before background subtraction.

Dim num_meta_rows As Integer
Dim num_meta_cols As Integer
Dim num_sub_rows As Integer
Dim num_sub_cols As Integer

Dim loop_meta_row As Integer
Dim loop_meta_col As Integer
Dim loop_sub_row As Integer
Dim loop_sub_col As Integer

Dim local_factor

Dim raw_sheet As Worksheet
Dim raw_range As Range
Dim local_sheet As Worksheet
Dim local_range As Range
Dim meta_range As Range

Set raw_sheet = ActiveSheet
Application.CutCopyMode = False
'copy the column of accession numbers
raw_sheet.Columns("A:A").Copy
' create a new sheet and name it "local"
Set local_sheet = Sheets.Add
local_sheet.Name = "local"
' paste in the column of accession numbers
local_sheet.Range("A1").Select
local_sheet.Paste
' count how many rows there are to calculate
acc_count = Range("A1", Range("A1").End(xlDown)).Rows.Count

```

```

' format the first row of the local sheet
With local_sheet.Rows("1:1")
    .HorizontalAlignment = xlGeneral
    .VerticalAlignment = xlBottom
    .WrapText = True
    .Orientation = 0
    .AddIndent = False
    .ShrinkToFit = False
    .MergeCells = False
End With

Set raw_range = raw_sheet.Range("G1")
Set local_range = local_sheet.Range("F1")
num_meta_rows = Application.WorksheetFunction.Max(raw_sheet.Range("B:B"))
num_meta_cols = Application.WorksheetFunction.Max(raw_sheet.Range("C:C"))
num_sub_rows = Application.WorksheetFunction.Max(raw_sheet.Range("D:D"))
num_sub_cols = Application.WorksheetFunction.Max(raw_sheet.Range("E:E"))
,
While raw_range.Value <> ""
    local_range.Value = _
        Replace(raw_range.Value, "- median_background", "Local corrected")
    ' format the numbers in this column
    local_range.EntireColumn.NumberFormat = "0.00"
    Set raw_range = raw_range.Offset(1, 0)
    Set local_range = local_range.Offset(1, 0)

    For loop_meta_row = 1 To num_meta_rows
        For loop_meta_col = 1 To num_meta_cols
            For loop_sub_row = 1 To num_sub_rows

                For loop_sub_col = 1 To num_sub_cols
                    Select Case loop_sub_row
                        Case 1
                            If loop_sub_col = 1 Then
                                local_factor =
Application.WorksheetFunction.Average _
                                (raw_range, raw_range.Offset(1, 0), _
                                raw_range.Offset(num_sub_cols, 0),
raw_range.Offset(num_sub_cols + 1, 0))
                            Else
                                local_factor =
Application.WorksheetFunction.Average _
                                (raw_range.Offset(-1, 0), raw_range,
raw_range.Offset(1, 0), _

```

```

        raw_range.Offset(num_sub_cols - 1, 0),
raw_range.Offset(num_sub_cols, 0), raw_range.Offset(num_sub_cols + 1, 0))
    End If
    Case num_sub_rows
        If loop_sub_col = num_sub_col Then
            local_factor =
Application.WorksheetFunction.Average _
                (raw_range.Offset(-1, 0), raw_range, _
                raw_range.Offset(-(num_sub_cols - 1), 0),
raw_range.Offset(-num_sub_cols, 0))
        Else
            local_factor =
Application.WorksheetFunction.Average _
                (raw_range.Offset(-1, 0), raw_range,
raw_range.Offset(1, 0), _
                raw_range.Offset(-(num_sub_cols - 1), 0),
raw_range.Offset(-num_sub_cols, 0), raw_range.Offset(-(num_sub_cols + 1), 0))
        End If
    Case Else
        Select Case loop_sub_col
            Case 1
                local_factor =
Application.WorksheetFunction.Average _
                    (raw_range, raw_range.Offset(1, 0), _
                    raw_range.Offset(num_sub_cols, 0),
raw_range.Offset(num_sub_cols + 1, 0), _
                    raw_range.Offset(-num_sub_cols, 0),
raw_range.Offset(-(num_sub_cols + 1), 0))
            Case num_sub_col
                local_factor =
Application.WorksheetFunction.Average _
                    (raw_range.Offset(-1, 0), raw_range, _
                    raw_range.Offset(num_sub_cols - 1, 0),
raw_range.Offset(num_sub_cols, 0), _
                    raw_range.Offset(-(num_sub_cols - 1),
0), raw_range.Offset(-num_sub_cols, 0))
            Case Else
                local_factor =
Application.WorksheetFunction.Average _
                    (raw_range.Offset(-1, 0), raw_range,
raw_range.Offset(1, 0), _
                    raw_range.Offset(num_sub_cols - 1, 0),
raw_range.Offset(num_sub_cols, 0), raw_range.Offset(num_sub_cols + 1, 0), _
                    raw_range.Offset(-(num_sub_cols - 1),
0), raw_range.Offset(-num_sub_cols, 0), raw_range.Offset(-(num_sub_cols + 1), 0))
        End Select 'COL

```

```

        End Select 'ROW

        local_range.Value = raw_range.Offset(0, -1).Value -
local_factor

        Set local_range = local_range.Offset(1, 0)
        Set raw_range = raw_range.Offset(1, 0)

        Next loop_sub_col
        Next loop_sub_row
        Next loop_meta_col
    Next loop_meta_row

    ' move over to the next set of columns
    Set raw_range = raw_sheet.Cells(1, raw_range.Column + 3)
    Set local_range = local_sheet.Cells(1, local_range.Column + 1)
Wend
End Sub

Private Function newWorksheetName(fullNameWS As String, shortNameWS As String) As String
    'Argument 1 = descriptive name of new worksheet, used in prompts
    'Argument 2 = default name of new worksheet, actual name in Excel
    'Creates a new worksheet named by the user, using Argument 2 as default
    'If the chosen name conflicts with an existing worksheet,
    ' user has choice of replacing (deleting) the existing worksheet
    ' or renaming the new worksheet

    'Returns the name of the new worksheet given by the user as String

    Dim myName As String

    Do
        myName = InputBox("Enter name of " & fullNameWS & " worksheet:", _
            "Add " & fullNameWS & " Worksheet", shortNameWS)
        If myName = "" Then End
        flag = True
        For i = 1 To Sheets.Count
            If Sheets(i).Name = myName Then
                response = MsgBox( _
                    prompt:="Do you want to replace the existing worksheet named
"" & myName & ""?", _
                    Buttons:=vbYesNoCancel + vbDefaultButton2, _
                    Title:="Replace Worksheet")
                Select Case response

```

```

        Case vbCancel
            'Cancel button selected
        End
    Case vbYes
        'Yes button selected
        Application.DisplayAlerts = False
        Sheets(i).Delete
        Application.DisplayAlerts = True
        flag = True
        Exit For
    Case vbNo
        'No button selected
        flag = False
        Exit For
    End Select
End If
Next i
Loop While flag <> True
newWorksheetName = myName
End Function

Private Sub Filter_Dupl()

    ' Filter_Dupl Macro
    ' Macro written by Matthew J. Rodland
    ' Last updated 7/17/2001 by Matthew J. Rodland

    ' Reads in data from active duplicate comparison worksheet and filters the data
    ' replacing values where any duplicate or const_fc flag is false with a null value.
    ' Creates and records this information on a new worksheet in the same workbook.

    Dim col_count As Integer
    Dim dupl_sheet As Worksheet
    Dim dest_sheet As Worksheet
    Dim dest_range As Range

    Dim row_count As Integer
    Dim column_count As Integer
    Dim dupl_address As String
    Dim avg_address As String
    Dim const_flag_address As String
    Dim const_FC_range As Range

```

```

Dim num_duplicates As Integer

Dim myName As String

'Create a new worksheet named by the user
Set dupl_sheet = ActiveSheet
myName = newWorksheetName("Filtered Duplicates", "dupl_data")
Set dest_sheet = Sheets.Add
dest_sheet.Name = myName

'Count rows and columns
row_count = dupl_sheet.Range("A1").End(xlDown).row
column_count = dupl_sheet.Range("A1").End(xlToRight).Column - 2 'exclude
accession and const_fc from count

'Copy accession column
dupl_sheet.Range(dupl_sheet.Cells(1, 1), dupl_sheet.Cells(row_count, 1)).Copy
dest_sheet.Paste
dest_sheet.Range("1:1").ColumnWidth = 9

'Find Const_FC flag, allowing for Grid Location to be on the worksheet
Set const_FC_range = dupl_sheet.Cells(1, column_count + 2)
num_duplicates = column_count \ 4
While const_FC_range <> "Const FC?"
    'Show error if Const_FC flag could not be found
    If const_FC_range.Column = 1 Then
        result = _
        MsgBox("Error - Could not find column labeled ""Const FC?"". " & _
            " Exiting automation script.", _
            Buttons:=vbOKOnly, _
            Title:="Error")
    End
End If

'Insert grid or other data to the right of duplicate averages
const_FC_range.EntireColumn.Copy
dest_sheet.Cells(1, num_duplicates + 1).EntireColumn.PasteSpecial
dest_sheet.Cells(1, num_duplicates + 1).EntireColumn.Insert
Set const_FC_range = const_FC_range.Offset(0, -1)
column_count = column_count - 1
Wend

'Address of Constant FC flag
const_FC_address = dupl_sheet.Name & "!" & _
    dupl_sheet.Cells(2, column_count + 2).Address( _

```

```

        RowAbsolute:=False, _
        ColumnAbsolute:=False)

'Loop for each pair of duplicates to filter
For iter = 1 To (column_count \ 4)
    'copy column header
    dupl_sheet.Cells(1, 1 + 4 * iter).Copy
    dest_sheet.Cells(1, 1 + iter).PasteSpecial
    'Address of Duplicate Flag
    flag_address = dupl_sheet.Name & "!" & _
        dupl_sheet.Cells(2, 4 * iter).Address( _
            RowAbsolute:=False, _
            ColumnAbsolute:=False)
    'Address of Duplicate Average
    avg_address = dupl_sheet.Name & "!" & _
        dupl_sheet.Cells(2, 1 + 4 * iter).Address( _
            RowAbsolute:=False, _
            ColumnAbsolute:=False)
    'Create formula for filtering duplicates
    dest_sheet.Cells(2, 1 + iter).Formula = _
        "=IF(AND(" & flag_address & "," & const_FC_address & "), " & _
        avg_address & ", "" "")"
    'Copy formula down the column
    dest_sheet.Cells(2, 1 + iter).Copy
    dest_sheet.Range(dest_sheet.Cells(3, 1 + iter), _
        dest_sheet.Cells(row_count, 1 + iter)).PasteSpecial
Next iter
dest_sheet.Cells.NumberFormat = "0.00"
Application.CutCopyMode = False
dest_sheet.Cells.Copy
dest_sheet.Range("A1").PasteSpecial xlPasteValues
dest_sheet.Range("A1").Select
End Sub

Private Sub Quality_Summary()

' Quality_Summary Macro
' Macro written by Matthew J. Rodland
' Last updated 7/17/2001 by Matthew J. Rodland

' This macro searches for the local corrected and duplicate comparison worksheets
' (only the local corrected is required), calculates quality control measures,

```

' and saves the report on a new worksheet in the same workbook.

```
Dim flag As Boolean
Dim quality_sheet As Worksheet
Dim local_sheet As Worksheet
Dim quality_name As String
Dim local_name As String
Dim local_range As Range
Dim quality_range As Range
Dim local_row_count As Integer
Dim dupl_row_count As Integer
Dim dupl_name As String
Dim dupl_sheet As Worksheet
Dim dupl_range As Range
Dim col_count As Integer

' create a new sheet and name it
quality_name = newWorksheetName("Quality Summary", "quality")
Set quality_sheet = Sheets.Add
quality_sheet.Name = quality_name

flag = False
local_name = "local"
Do
    For iter = 1 To Sheets.Count
        If Sheets(iter).Name = local_name Then
            flag = True
            Set local_sheet = Sheets(iter)
        End If
    Next iter
    If flag <> True Then
        local_name = InputBox( _
            prompt:="Enter the name of the Local corrected worksheet ", _
            Title:="Find Local Corrected Worksheet", _
            Default:="local")
        If local_name = "" Then End
    End If
Loop While flag <> True

flag = False
dupl_name = "dupl_compare"
Do
```

```

For iter = 1 To Sheets.Count
    If Sheets(iter).Name = dupl_name Then
        flag = True
        Set dupl_sheet = Sheets(iter)
    End If
Next iter
If flag <> True Then
    dupl_name = InputBox( _
        prompt:="Enter the name of the Duplicate Comparison worksheet. " &
        "If this worksheet does not exist, click the 'Cancel' button.",
        Title:="Find Duplicate Comparison Worksheet", _
        Default:="dupl_compare")
    If dupl_name = "" Then Exit Do
End If
Loop While flag <> True

'Obtain header list from local corrected sheet
local_sheet.Range("1:1").Copy
quality_sheet.Range("1:1").PasteSpecial xlPasteValues
'Strip off any meta-grid or position info from quality summary
col_count = quality_sheet.Range("A1").End(xlToRight).Column
For Each iter In quality_sheet.Range(quality_sheet.Cells(1, 1),
quality_sheet.Cells(1, col_count))
    If iter.Value = "meta_row" _
    Or iter.Value = "meta_col" _
    Or iter.Value = "sub_row" _
    Or iter.Value = "sub_col" _
    Or iter.Value = "position" Then
        iter.Value = ""
        col_count = col_count - 1
    End If
Next iter

quality_sheet.Range("A1").Value = ""
quality_sheet.Range("A2").Value = "20% Slidewise Trimmed Mean:"
quality_sheet.Range("A3").Value = "Slidewise Mean:"
quality_sheet.Range("A4").Value = "Slidewise Median:"
quality_sheet.Range("A5").Value = "Slidewise Mode:"
quality_sheet.Range("A6").Value = "% Above Saturation (55k):"
quality_sheet.Range("A7").Value = "% Below Floor (500):"
If flag = True Then quality_sheet.Range("A10").Value = "% Bad Duplicates:"

```

```

quality_sheet.Cells.NumberFormat = "0.00"
quality_sheet.Range("A:A").ColumnWidth = 25.14
quality_sheet.Range("1:1").WrapText = True

local_row_count = local_sheet.Range("A1").End(xlDown).row
'Find computed trimmed mean values from local corrected worksheet
Set local_range = local_sheet.Range("A1").End(xlDown).Offset(3, 1)
Range(local_range, local_range.End(xlToRight)).Copy
quality_sheet.Range("B2").PasteSpecial xlPasteValues

'Compute Slidewise Mean, Median, Mode
For iter = 2 To col_count
quality_sheet.Cells(3, iter).FormulaR1C1 = _
    "=average(" & local_sheet.Name & "!R2C" & iter & ":R" & local_row_count &
"C" & iter & ")"
quality_sheet.Cells(4, iter).FormulaR1C1 = _
    "=median(" & local_sheet.Name & "!R2C" & iter & ":R" & local_row_count &
"C" & iter & ")"
quality_sheet.Cells(5, iter).FormulaR1C1 = _
    "=mode(" & local_sheet.Name & "!R2C" & iter & ":R" & local_row_count & "C"
& iter & ")"
quality_sheet.Cells(6, iter).FormulaR1C1 = _
    "=countif(" & local_sheet.Name & "!R2C" & iter & ":R" & local_row_count &
"C" & iter & ", ">55000")/" & local_row_count - 1
quality_sheet.Cells(6, iter).NumberFormat = "0%"
quality_sheet.Cells(7, iter).FormulaR1C1 = _
    "=countif(" & local_sheet.Name & "!R2C" & iter & ":R" & local_row_count &
"C" & iter & ", "<500")/" & local_row_count - 1
quality_sheet.Cells(7, iter).NumberFormat = "0%"
Set quality_range = quality_sheet.Cells(10, 2)
Next iter

If flag = True Then
    Set dupl_range = dupl_sheet.Range("A1").End(xlDown).Offset(2, 3)
    quality_sheet.Range("9:9").WrapText = True
    dupl_row_count = dupl_sheet.Range("A1").End(xlDown).row
    While dupl_range.Value <> ""
        dupl_sheet.Cells(1, dupl_range.Column + 1).Copy
        quality_sheet.Cells(9, quality_range.Column).PasteSpecial xlPasteValues
        quality_range.Value = dupl_range.Value / (dupl_row_count - 1)
        quality_range.NumberFormat = "0%"
        Set dupl_range = dupl_range.Offset(0, 4)
        Set quality_range = quality_range.Offset(0, 1)
    End While
End If

```

```

        Wend
    End If
End Sub

Private Sub Graph_Duplicates()

' Graph_Duplicates Macro
' Macro written by Matthew J. Rodland
' Last updated 7/17/2001 by Matthew J. Rodland

' This macro creates a new worksheet on the same workbook after searching
' for the duplicate comparison worksheet and creating duplicate graphs
' for each duplicate pair.

    Dim graph_sheet As Worksheet
    Dim graph_range As Range
    Dim dupl_sheet As Worksheet
    Dim dupl_range As Range
    Dim dupl_name As String
    Dim col_iter As Integer
    Dim row_count As Integer
    Dim series_range As String
    Dim iter As Integer
    Dim flag As Boolean

    'Locate the duplicate comparison sheet
    flag = False
    dupl_name = "dupl_compare"
    Do
        For iter = 1 To Sheets.Count
            If Sheets(iter).Name = dupl_name Then
                flag = True
                Set dupl_sheet = Sheets(iter)
            End If
        Next iter
        'If duplicate comparison sheet not found, ask the user
        If flag <> True Then
            dupl_name = InputBox( _
                prompt:="Enter the name of the Duplicate Comparison worksheet. " &

```

```

        "If this worksheet does not exist, click the 'Cancel' button.",

Title:="Find Duplicate Comparison Worksheet", _
Default:="dupl_compare")
    If dupl_name = "" Then End
End If
Loop While flag <> True

row_count = dupl_sheet.Range("A1").End(xlDown).row
col_count = dupl_sheet.Range("1:1").Find("Const FC?",
LookIn:=xlValues).Offset(0, -4).Column
Set dupl_range = dupl_sheet.Range(dupl_sheet.Cells(2, 2),
dupl_sheet.Cells(row_count, 2))

'Create a new worksheet named by the user
myName = newWorksheetName("Duplicate Graphs", "graphs")
Set graph_sheet = Sheets.Add
graph_sheet.Name = myName

'add X-fold range bars
Range("A2").Select
ActiveCell.Value = "0.01"
Range("A3").Select
ActiveCell.Value = "0.1"
Range("A4").Select
ActiveCell.Value = "1"
Range("A5").Select
ActiveCell.Value = "10"
Range("B1").Select
ActiveCell.FormulaR1C1 = "=Dupl_FC"
Range("B2").Select
ActiveCell.FormulaR1C1 = "=RC1/R1C2"
Range("B2").Select
Selection.Copy
Range("B2:B5").Select
ActiveSheet.Paste
Range("C2").Select
ActiveSheet.Paste
Application.CutCopyMode = False
ActiveCell.FormulaR1C1 = "=RC1*R1C2"
Range("C2").Select
Selection.Copy
Range("C2:C5").Select

```

```

ActiveSheet.Paste
Range("A1").Select

'create graph for each duplicate pair
iter = 1
col_iter = 2
chart_iter = 1
Do
    Charts.Add
    ActiveChart.ChartType = xlXYScatter
    ActiveChart.SetSourceData Source:=Sheets(dupl_sheet.Name).Range("A1")
    ActiveChart.SeriesCollection.NewSeries
    series_range = "=" & dupl_sheet.Name & "!R2C" & col_iter & ":R" & row_count
    & "C" & col_iter
    ActiveChart.SeriesCollection(1).XValues = series_range
    series_range = "=" & dupl_sheet.Name & "!R2C" & col_iter + 1 & ":R" &
row_count & "C" & col_iter + 1
    ActiveChart.SeriesCollection(1).Values = series_range
    ActiveChart.SeriesCollection.NewSeries
    ActiveChart.SeriesCollection(2).XValues = "=" & myName & "!R2C1:R5C1"
    ActiveChart.SeriesCollection(2).Values = "=" & myName & "!R2C2:R5C2"
    ActiveChart.SeriesCollection.NewSeries
    ActiveChart.SeriesCollection(3).XValues = "=" & myName & "!R2C1:R5C1"
    ActiveChart.SeriesCollection(3).Values = "=" & myName & "!R2C3:R5C3"
    ActiveChart.SeriesCollection(2).Trendlines.Add(Type:=xlLinear, Forward:=0,
-
    Backward:=0, DisplayEquation:=False, DisplayRSquared:=False).Select
    ActiveChart.SeriesCollection(3).Trendlines.Add(Type:=xlLinear, Forward:=0,
-
    Backward:=0, DisplayEquation:=False, DisplayRSquared:=False).Select

    ActiveChart.Location Where:=xlLocationAsObject, Name:=graph_sheet.Name
    With ActiveChart
        .HasTitle = True
        .ChartTitle.Characters.Text = _
            Replace(dupl_sheet.Cells(1, col_iter + 3).Value, "Average",
"Duplicates")
        .Axes(xlCategory, xlPrimary).HasTitle = True
        .Axes(xlCategory, xlPrimary).AxisTitle.Characters.Text = _
            dupl_sheet.Cells(1, col_iter).Value
        .Axes(xlValue, xlPrimary).HasTitle = True
        .Axes(xlValue, xlPrimary).AxisTitle.Characters.Text = _
            dupl_sheet.Cells(1, col_iter + 1).Value

```

```

End With
With ActiveChart.Axes(xlCategory)
    .HasMajorGridlines = True
    .HasMinorGridlines = False
End With
With ActiveChart.Axes(xlValue)
    .HasMajorGridlines = True
    .HasMinorGridlines = False
End With
ActiveChart.HasLegend = False
ActiveSheet.Shapes("Chart " & chart_iter).ScaleWidth 0.98, msoFalse, _
    msoScaleFromBottomRight
ActiveSheet.Shapes("Chart " & chart_iter).ScaleHeight 1.32, msoFalse, _
    msoScaleFromBottomRight
ActiveSheet.Shapes("Chart " & chart_iter).ScaleWidth 1.07, msoFalse,
msoScaleFromTopLeft
ActiveSheet.Shapes("Chart " & chart_iter).ScaleHeight 1.16, msoFalse,
msoScaleFromTopLeft
increment_offset = 192 * (iter - 1)
ActiveSheet.Shapes("Chart " & chart_iter).IncrementLeft increment_offset
'set x and y axis to log scale
ActiveChart.Axes(xlCategory).ScaleType = xlLogarithmic
ActiveChart.Axes(xlValue).ScaleType = xlLogarithmic
'set data point size to smallest
ActiveChart.SeriesCollection(1).MarkerSize = 2

iter = iter + 2
col_iter = col_iter + 4
chart_iter = chart_iter + 1
Loop While col_iter <= col_count

End Sub

```