2008

Probe Level Analysis of Affymetrix Microarray Data

Richard Ellis Kennedy
Virginia Commonwealth University

Probe Level Analysis of Affymetrix Microarray Data

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

by Richard Ellis Kennedy

Bachelor of Arts in Computer Science, 1990
University of Mississippi, Oxford, Mississippi

Doctor of Medicine, 1994
University of Mississippi Medical Center, Jackson, Mississippi

Director: Kellie J. Archer, Ph.D.
Assistant Professor
Department of Biostatistics

Virginia Commonwealth University
Richmond, Virginia
May 2008

Acknowledgements

A project of this magnitude represents the culmination of many months of effort, but represents only a small portion of my time in graduate school. In the past six years, I have met many new friends, begun a new career, matured in my marriage, and seen the birth of my two children. I am grateful to all of the people who have made this experience so rich and rewarding.

My advisor, Dr. Kellie Archer, has been a wonderful mentor during my studies and the completion of my dissertation. Her intellectual curiosity and breadth of knowledge was, and will continue to be, an excellent example for me to follow in my own work. She has also been steadfastly supportive of my pursuits and a source of encouragement during the difficult times that are a part of graduate school. Without her guidance, this dissertation would still be just an interesting idea that I had come across during the course of my studies.

My remaining committee members have also been excellent. Dr. Ramesh has never ceased to give of his time, both inside and outside of the classroom, to help with my learning; he is a teacher in the truest sense of the word. Dr. Best has always provided a sharp eye for details, and my dissertation would not be nearly so polished and eloquent without his involvement. Dr. Miles and Dr. Zhao have both ensured that I have a good understanding of the biological principles that are so critical to work in this area of biostatistics. I thank you all for your influence on my education and my career.

All of this would not have been possible without the support of my family through the years. My mother and father instilled a lifelong commitment to learning through their example and made sure that ample educational opportunities were available to me. It is only now, as a new parent, that I truly begin to appreciate all of the sacrifices that they have made. My wife and two sons have patiently endured the long process of graduate school with me, and given their love and support throughout. I am glad to have had you all with me during this journey; I could not have done it without you.

Table of Contents

List of Tables v

List of Figures vi

List of Symbols x

Abstract xiii

1 Introduction 1
- 1.1 Overview of Microarray Technology . . . 1
  - 1.1.1 cDNA Array Design . . . 1
  - 1.1.2 Oligonucleotide Array Design . . . 3
  - 1.1.3 Comparison of cDNA and Oligonucleotide Array Technologies . . . 7
- 1.2 Experimental Design . . . 8
  - 1.2.1 Introduction . . . 8
  - 1.2.2 Experimental Design of cDNA Microarray Studies . . . 9
  - 1.2.3 Experimental Design of Oligonucleotide Microarray Studies . . . 10
  - 1.2.4 Notational Conventions . . . 10
- 1.3 Image Analysis . . . 11
- 1.4 Early Methods for Computing GeneChip Expression . . . 12
- 1.5 Current Methods for Computing GeneChip Expression . . . 13
  - 1.5.1 Introduction . . . 13
  - 1.5.2 Pre-processing . . . 14
  - 1.5.3 dChip . . . 15
  - 1.5.4 MAS5 . . . 17
  - 1.5.5 RMA . . . 20
- 1.6 Hypothesis Testing . . . 22
- 1.7 Mixed Effects Models for Microarrays . . . 26
  - 1.7.1 Introduction . . . 26
  - 1.7.2 Advantages of the Mixed Effects Model . . . 29
  - 1.7.3 Implementation . . . 30
- 1.8 Review of Probe-Level Analysis . . . 31
  - 1.8.1 Logit-t . . . 32
  - 1.8.2 Multi-mgMOS . . . 35
  - 1.8.3 Probe-level ANOVA . . . 36
- 1.9 Summary of Current Research . . . 38

2 The S-Score Algorithm **40**
2.1 Development . . . 40
2.2 Implementation . . . 44
2.3 Performance Assessment . . . 44
2.4 Limitations . . . 54

3 The Random Variance Model Algorithm **56**
3.1 Development . . . 56
3.2 Implementation . . . 66
3.3 Performance Assessment . . . 68
3.4 Limitations . . . 69

4 Extending Error Models **71**
4.1 Introduction . . . 71
4.2 Nonparametric Error Models . . . 72
4.3 Parametric Error Models for Variances Only . . . 78
4.3.1 Prerequisite Matrix Algebra . . . 78
4.3.1.1 Basic Results and Definitions . . . 78
4.3.1.2 Matrix Decompositions . . . 81
4.3.1.3 Kronecker Product . . . 82
4.3.1.4 Vec and Vech Operators . . . 84
4.3.1.5 Patterned Matrices: Elimination, Duplication, Commutation, and Symmetrizer Matrices . . . 87
4.3.1.6 Jacobian Determinant . . . 96
4.3.1.7 Matrix Derivative and Differential . . . 97
4.3.2 Results for Derivatives and Jacobian Matrices . . . 103
4.3.2.1 Scalar Functions of a Matrix . . . 103
4.3.2.2 Matrix Functions of a Matrix . . . 109
4.3.3 Probability Distributions . . . 116
4.3.3.1 Wishart and Related Distributions . . . 117
4.3.3.2 Multivariate Beta and Related Distributions . . . 123
4.3.4 Development of the Multivariate Random Variance Model . . . 130
4.3.5 Modified Random Variance Model for Singular Covariance Matrices . . . 153
4.4 Parametric Error Models for Intensities . . . 159

5 Performance Assessment **160**
5.1 Introduction . . . 160
5.2 Overview of Spike-in Studies . . . 161
5.3 Methods for Comparisons . . . 167
5.3.1 Data . . . 167
5.3.2 Data Processing . . . 169
5.3.3 Selection of Baseline . . . 170
5.3.4 Quality Assessment and Data Integrity Checks . . . 170
5.3.5 Statistical Analysis . . . 171
5.4 Results . . . 178
5.4.1 Quality Assessment . . . 178

5.4.2 Statistical Analysis 182

6 Discussion **192**
6.1 Overview 192
6.2 Quality of Spike-In Datasets 193
6.3 The S-Score Algorithm 194
6.4 The Random Variance Model 196

7 Future Research **203**
7.1 Spike-In Datasets 203
7.2 The S-Score Algorithm 205
7.3 The Random Variance Model 207

8 Conclusions **213**

Bibliography **215**

A Source Code Listings **234**
A.1 Quality Control Assessment of the Choe *et al.* Spike-in Dataset 235
A.2 Logit-T Analysis of Spike-In Datasets 245
A.3 mmgMOS Analysis of Spike-In Datasets 259
A.4 Multichip S-Score Analysis of Spike-In Datasets 272
A.5 Pooled S-Score Analysis of Spike-In Datasets 285
A.6 RMA Analysis of Spike-In Datasets 309
A.7 RVM Analysis of Spike-In Datasets 322

B Quality Assessment Plots for All Datasets **352**

C Annotation Data for Omitted Probesets **369**

List of Tables

1.1 Overview of Probeset Summary Methods for Affymetrix Data. 14
1.2 Outcomes from N Simultaneous Hypothesis Tests 24

2.1 Concentration Data for the GeneLogic Dilution Dataset 48
2.2 Concentration Data for the GeneLogic AML Latin Square Dataset 49
2.3 Number and Proportion of GeneLogic Spike-In Clones Detected 54

5.1 Probeset Groupings for the Affymetrix U95 Spike-In Study 162
5.2 Concentration Data for the Affymetrix U95 Spike-In Study 163
5.3 Probeset Groupings for the Affymetrix U133 Spike-In Study 164
5.4 Concentration Data for the Affymetrix U133 Spike-In Study 165
5.5 Concentration Data for the GeneLogic Tonsil Latin Square Dataset 166
5.6 Concentration Data for the Choe *et al.* Golden Spike Dataset 168
5.7 Clone and Pool Assignments for the Choe *et al.* Golden Spike Dataset Using Indirect Mappings 186
5.8 Number of True Positives in Analysis of the GeneLogic Dilution Dataset 187
5.9 Number of False Positives in Analysis of the GeneLogic Dilution Dataset 187
5.10 Number of True Negatives in Analysis of the GeneLogic Dilution Dataset 187
5.11 Number of False Negatives in Analysis of the GeneLogic Dilution Dataset 188
5.12 Statistical Significance for True Positives in the GeneLogic Dilution Analysis . . 188
5.13 Statistical Analysis of the Affymetrix U95 Latin Square Dataset 189
5.14 Statistical Significance for the Affymetrix U95 Latin Square Analysis 190
5.15 Statistical Analysis of the Affymetrix U133 Latin Square Dataset 191
5.16 Statistical Significance for the Affymetrix U133 Latin Square Analysis 191

C.1 Probesets Omitted from the Affymetrix U95 Latin Square and GeneLogic Dilution Analysis 370
C.2 Probesets Omitted from the Affymetrix U95 Latin Square and GeneLogic Dilution Analysis 371
C.3 Probesets Omitted from the Affymetrix U133 Latin Square Analysis 372

List of Figures

1.1 Overview of cDNA Array Synthesis 2
1.2 Overview of GeneChip Synthesis 4
1.3 Photolithographic mask 4
1.4 Probe Synthesis 5
1.5 A Completed GeneChip 5
1.6 Structure of a GeneChip 6

2.1 Comparison of S-Score and RMA 51
2.2 Comparison of S-Score and dChip 52
2.3 Comparison of S-Score and MAS5 53

5.1 GeneLogic Dilution Quality 179
5.2 GeneLogic AML Latin Square Quality 180
5.3 GeneLogic Tonsil Latin Square Quality 181
5.4 Affymetrix U95 Latin Square Quality 183
5.5 Affymetrix U133 Latin Square Quality 184
5.6 Choe Golden Spike Quality 185

B.1 GeneLogic Dilution Quality, Experiments 1–4 353
B.1 GeneLogic Dilution Quality, Experiments 5–8 354
B.1 GeneLogic Dilution Quality, Experiments 9–12 355
B.1 GeneLogic Dilution Quality, Experiments 13–16 356
B.1 GeneLogic Dilution Quality, Experiments 17–20 357
B.1 GeneLogic Dilution Quality, Experiments 21–24 358
B.1 GeneLogic Dilution Quality, Experiments 25–26 359
B.2 Affymetrix U95 Latin Square Quality, Experiments 1–4 360
B.2 Affymetrix U95 Latin Square Quality, Experiments 5–8 361
B.2 Affymetrix U95 Latin Square Quality, Experiments 9–12 362
B.2 Affymetrix U95 Latin Square Quality, Experiments 13–16 363
B.2 Affymetrix U95 Latin Square Quality, Experiments 17–20 364
B.3 Affymetrix U133 Latin Square Quality, Experiments 1–4 365
B.3 Affymetrix U133 Latin Square Quality, Experiments 5–8 366
B.3 Affymetrix U133 Latin Square Quality, Experiments 9–12 367
B.3 Affymetrix U133 Latin Square Quality, Experiments 13–14 368

List of Symbols

P	probes or probe pairs	10
S	probesets	10
M	microarrays or arrays or chips	10
C	experimental condition or class	10
D	dye	10
G	gene	10
L	cell line	10
N	number of elements for a given level or variable	10
$AvgDiff$	average difference	12
MM	mismatch probe intensity	12
PM	perfect match probe intensity	12
IS	set of rank-invariant probes	16
i	index or counter	16
δ	threshold for decision-making	16
μ	mean	16
ϕ	proportion of specific binding for MM probes	16
ε	random error term	16
y	intensity values for modeling (PM or $PM - MM$)	16
$T_{bi}(\cdot)$	Tukey's biweight function	17
n	number of observations *or* degrees of freedom	17
x	observation *or* scalar input for function	17
$w_{bi}(\cdot)$	weight function for Tukey's biweight	18
$u(\cdot)$	uniform distance measure for Tukey's biweight	18
τ_1	MAS5 tuning constant	18
τ_2	MAS5 tuning constant	18
bZ	MAS5 zone background	18
(x, y)	position coordinates	18
$w_i(\cdot)$	MAS5 weight function	18
$d(\cdot, \cdot)$	distance function	18
τ_{sm}	MAS5 smoothing constant	18
SB	MAS5 specific background	19
IM	idealized mismatch intensity	19
τ_c	MAS5 contrast tuning constant	19
τ_s	MAS5 scale tuning constant	19
LV	MAS5 log expression value	19

SF	MAS5 scale factor	20
EV	MAS5 expression value	20
tg	target intensity value for scale normalization	20
sg	probe-specific signal	20
bg	nonspecific background signal	20
σ^2	variance	20
$\boldsymbol{Y}$	matrix of intensities	21
n	number of items in subset *or* degrees of freedom	24
A	number of hypotheses accepted	24
R	number of hypotheses rejected *or* coefficient of variation	24
α	type I error rate *or* normalization constant	24
$FWER$	family-wise error rate	24
FDR	false discovery rate	25
H_T	number of true hypotheses	25
H_F	number of false hypotheses	25
H	hypothesis	25
T	test statistic	25
p	p-value	25
$F(\cdot)$	cumulative distribution function	25
j	index or counter	25
q	false discovery rate level of error	25
(MG)	array by gene interaction	26
(CG)	condition by gene interaction	26
$\boldsymbol{x}$	design vector	26
$\boldsymbol{X}$	design matrix *or* matrix input for function	26
(CM)	array by condition interaction	27
$\hat{\varepsilon}$	residuals	27
ψ	random error term	27
(MD)	array by dye interaction	28
(LC)	cell line by condition interaction	28
(LP)	cell line by probe interaction	28
(CP)	condition by probe interaction	28
k_f	rate constant for forward reaction	32
k_r	rate constant for reverse reaction	32
to	total amount of probe for binding	33
sc	scale parameter for mmgMOS	35
sh	shape hyperparameter for mmgMOS	35
ch	scale hyperparameter for mmgMOS	35
$\Psi(\cdot)$	digamma function	36
$\Psi'(\cdot)$	first derivative of the digamma function	36
bv	background variance (noise)	40
γ	fractional multiplicative error for intensity-dependent variance	40
Z	Z (normal or approximately normal) test statistic	40
$\boldsymbol{\Phi}(\cdot)$	standard normal cumulative distribution function	41
$\boldsymbol{\beta}$	vector of coefficients	56

k	number of parameters in full model *or* rank of matrix	56
a	scalar hyperparameter for random variance model	57
b	scalar hyperparameter for random variance model	57
r	number of parameters in reduced model	57
$\widehat{\boldsymbol{\beta}}$	maximum likelihood estimate of coefficients under null hypothesis	58
$\widehat{\widehat{\boldsymbol{\beta}}}$	maximum likelihood estimate of coefficients under alternative hypothesis	58
$\boldsymbol{X}_\omega$	design matrix for reduced model	58
$\widehat{\widehat{SS}}$	sum of squares under null hypothesis	58
$\widehat{SS}$	sum of squares under alternative hypothesis	58
$\widetilde{SS}$	sum of squares in random variance model	58
z	precision (inverse variance) parameter	59
$\mathbf{0}$	vector or matrix of zeros	60
$\widehat{\sigma}^2$	estimated variance (mean squared error)	63
$\widetilde{\sigma}^2$	estimated variance (mean squared error) in random variance model	63
K	number of free parameters	63
u	transformation variable for random variance model	65
v	transformation variable for random variance model	65
$\widehat{\sigma}^2_{pooled}$	pooled sample variance under null hypothesis	67
t	Student's t test statistic	67
$\widetilde{\sigma}^2_{pooled}$	pooled sample variance in random variance model	67
$\widetilde{t}$	modified t test statistic in random variance model	67
F	Snedecor's F test statistic	67
$\widetilde{F}$	modified F test statistic in random variance model	67
$\overline{bv}$	SScore mean background	72
$\overline{y}$	SScore mean intensity	73
lm	difference in log intensity between conditions	76
$\overline{y}$	average log intensity of conditions	76
Q	quantiles	76
O	groups	76
ξ	median of observations	76
σ	variance of observations	76
$\varphi(\cdot)$	permutation function	80
cf	cofactor of a matrix	80
ζ	scalar constant	81
λ	eigenvalue	81
$\boldsymbol{e}$	eigenvector	81
$\boldsymbol{L}$	diagonal matrix of eigenvalues	81
$\boldsymbol{E}$	matrix of eigenvectors	81
$\boldsymbol{I}_p$	identity matrix	81
$\boldsymbol{X}^{1/2}$	matrix square root	82
$\boldsymbol{X}^*$	lower triangular decomposition	82
$\boldsymbol{v}$	vector	85
$\boldsymbol{L}_p$	elimination matrix	87
$\boldsymbol{D}_p$	duplication matrix	88
$\boldsymbol{K}_p$	commutation matrix	88

$\boldsymbol{N}_p$	symmetrizer matrix	88
h	limit constant for derivative	97
$r(\cdot)$	scalar remainder function	97
E	set of matrices on which a function is defined	100
$B(\cdot,\cdot)$	ball (neighborhood) about a matrix	100
rd	radius	100
$\boldsymbol{U}$	matrix within the neighborhood of a specified matrix	100
$\boldsymbol{A}$	(first) derivative matrix	100
$\boldsymbol{R}_c$	matrix remainder function	100
$\boldsymbol{G}(\cdot)$	matrix-valued function	101
$\boldsymbol{H}(\cdot)$	matrix-valued function	101
$\boldsymbol{C}$	constant matrix	101
η	eigenvalues	113
$\boldsymbol{Q}$	transformation matrix for singular matrices	115
$\boldsymbol{M}$	random matrix	117
$\Gamma_p(\cdot)$	multivariate gamma function	117
$\boldsymbol{\Xi}$	location parameter matrix	125
$\boldsymbol{\Omega}$	scale parameter matrix	125
Λ	Wilk's lambda statistic	129
$\boldsymbol{B}$	matrix hyperparameter for random variance model	130
$\boldsymbol{Z}$	precision (inverse covariance) matrix in random variance model	132
$\boldsymbol{O}$	intermediate result of matrix calculations	136
$\widehat{\boldsymbol{\Sigma}}$	sums of squares and crossproducts under alternative hypothesis	138
$\widehat{\widehat{\boldsymbol{\Sigma}}}$	sums of squares and crossproducts under null hypothesis	139
$\widetilde{\boldsymbol{SS}}$	sums of squares and crossproducts in random variance model	139
$\boldsymbol{U}$	transformed matrix in random variance model	141
$\boldsymbol{V}$	transformed matrix in random variance model	141

List of Acronyms

AI Adaptive Interval

AML acute myeloid leukemia

ANOVA analysis of variance

AUC area under the curve

CDF cumulative distribution function

cDNA complementary deoxyribonucleic acid

CLT Central Limit Theorem

cRNA complementary ribonucleic acid

DNA deoxyribonucleic acid

FDR false discovery rate

FWER family-wise error rate

GCOS GeneChip Operating Software

GLM general linear model

ISH *in situ* hybridization

IVT *in vitro* transcription

LPE local pooled error

MAD median absolute deviation

MANOVA multivariate analysis of variance

MAQC MicroArray Quality Control

MAS4 Microarray Suite version 4.0

MAS5 Microarray Suite version 5.0

MBEI model-based expression index

MLE maximum likelihood estimate

MM mismatch

mmgMOS multichip modified gamma model of oligonucleotide signal

mRNA messenger ribonucleic acid

PCR polymerase chain reaction

PRD proportional rank difference

PM perfect match

PPV positive predictive value

RIR rank invariant resampling

RMA robust multi-chip average

RNA ribonucleic acid

ROC receiver operating characteristic

RVM Random Variance Model

SAM Significance Analysis of Microarrays

SDT Statistical (or Standard) Difference Threshold

SF Scale Factor

TIFF tagged image file format

Abstract

PROBE LEVEL ANALYSIS OF AFFYMETRIX MICROARRAY DATA

By Richard Ellis Kennedy, Ph.D.

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2008

Major Director: Kellie J. Archer, Ph.D.
Assistant Professor
Department of Biostatistics

The analysis of Affymetrix GeneChip® data is a complex, multistep process. Most often, methods condense the multiple probe level intensities into single probeset level measures (such as robust multi-chip average (RMA), dChip and Microarray Suite version 5.0 (MAS5)), which are then followed by application of statistical tests to determine which genes are differentially expressed. An alternative approach is a probe-level analysis, which tests for differential expression directly using the probe-level data. Probe-level models offer the potential advantage of more accurately capturing sources of variation in microarray experiments. However, this has not been thoroughly investigated, since current research efforts have largely focused on the development of improved expression summary methods. This research project will review current approaches to analysis of probe-level data and discuss extensions of two examples, the S-Score and the Random Variance

Model (RVM). The S-Score is a probe-level algorithm based on an error model in which the detected signal is proportional to the probe pair signal for highly expressed genes, but approaches a background level (rather than 0) for genes with low levels of expression. Initial results with the S-Score have been promising, but the method has been limited to two-chip comparisons. This project presents extensions to the S-Score that permit comparisons of multiple chips and "borrowing" of information across probes to increase statistical power. The RVM is a probeset-level algorithm that models the variance of the probeset intensities as a random sample from a common distribution to "borrow" information across genes. This project presents extensions to the RVM for probe-level data, using multivariate statistical theory to model the covariance among probes in a probeset. Both of these methods show the advantages of probe-level, rather than probeset-level, analysis in detecting differential gene expression for Affymetrix GeneChip data. Future research will focus on refining the probe-level models of both the S-Score and RVM algorithms to increase the sensitivity and specificity of microarray experiments.

Chapter 1

Introduction

1.1 Overview of Microarray Technology

The development of microarray chips has had a profound impact on the design of gene expression studies (Shi *et al.*, 2006). Microarrays allow the large-scale analysis of expression changes across thousands of genes at once, in contrast to previous small-scale methods that were restricted to examining the expression of only a few genes at a time. Advances in technology have led to chips that are essentially capable of analyzing expression values across the entire genome simultaneously (Dalma-Weiszhausz *et al.*, 2006).

1.1.1 cDNA Array Design

Microarrays are broadly divided into two categories, custom spotted or complementary deoxyribonucleic acid (cDNA) arrays and oligonucleotide arrays (Nguyen *et al.*, 2002). The former are designed by synthesizing complementary probes to messenger ribonucleic acid (mRNA) obtained from biological samples using reverse transcription (Schena *et al.*, 1995), as shown in Figure 1.1. For custom spotted arrays, segments from genes of interest are amplified using the polymerase chain reaction (PCR) to attain sufficient quantities of deoxyribonucleic acid (DNA). These DNA fragments are then placed on glass microscope slides or a similar substrate in a high-density grid pattern using a robotic pin arrayer or inkjet printer. The instrumentation is relatively inexpensive compared to that needed for oligonucleotide chips, so cDNA arrays may be produced by local

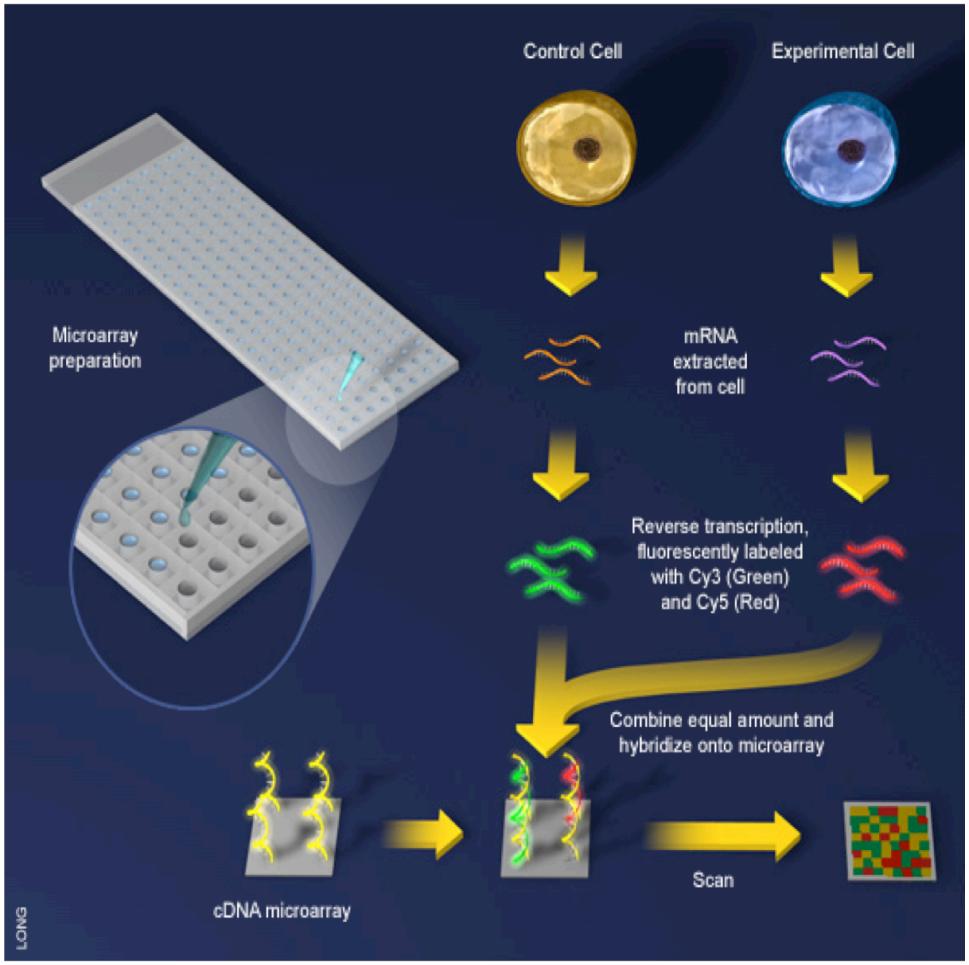

Figure 1.1: Overview of cDNA Array synthesis. DNA fragments are created using PCR and bound to glass microscope slides. These DNA fragments hybridize to complementary targets from samples. Image courtesy of Science Creative Quarterly (http://scq.ubc.ca), Jiang Long, artist.

laboratories or obtained commercially (Hardiman, 2004; Hager, 2006). Next, mRNA from two samples, one representing the experimental condition and one representing the control condition, are extracted and reverse transcribed into cDNA. The cDNA is labeled with fluorescent dye; by convention, Cyanine 5 (Cy5, a red fluorescent dye) is used for the experimental condition and Cyanine 3 (Cy3, a green fluorescent dye) is used for the control condition. The two samples are then mixed in equal proportion and hybridized to the array. The array is then imaged using a fluorescent scanner, with separate images collected for the red and green channels. The relative signal intensity between the two channels is assumed to be proportional to the relative abundance of the gene sequence of interest in the two samples (Duggan *et al.*, 1999).

1.1.2 Oligonucleotide Array Design

Oligonucleotide microarrays are designed by chemically synthesizing small nucleotide fragments with a predefined sequence (Lipshutz *et al.*, 1999). The Affymetrix GeneChip® oligonucleotide microarray is one of the most widely used and best standardized platforms for large-scale analysis of gene expression data (Wu *et al.*, 2004), although other platforms are available (Fan *et al.*, 2006; Wolber *et al.*, 2006; Shi *et al.*, 2006; Singh-Gasson *et al.*, 1999). An overview of GeneChip manufacture can be found in Dalma-Weiszhausz *et al.* (2006) and is shown in Figure 1.2. The basic unit for GeneChip design is a *probe*, a single 25-mer intended to hybridize with a specific transcript of a specific gene, which is called the target sequence. The target sequence reported by Affymetrix represents a "consensus" sequence for a particular gene, using information derived from public repositories and other sources (Alberts *et al.*, 2007). Target sequences are intended to be unique for a particular gene; however, it is recognized that this does not hold for a number of targets (Stalteri and Harrison, 2007; Gautier *et al.*, 2004). A complete list of target sequences for current and past varieties of GeneChips is available at the Affymetrix website (http://www.affymetrix.com).

GeneChips containing complementary probes to the target sequences are constructed using photolithography (Dalma-Weiszhausz *et al.*, 2006). For this process, quartz wafers are modified with silane material to allow covalent attachment of nucleosides to the chip. A photolithographic mask containing windows to permit the transmission of ultraviolet light is then aligned with the wafer. The windows on the mask are spatially distributed over the mask to correspond to the sequence for each probe. The application of near-ultraviolet light activates the exposed (windowed) areas of the wafer for nucleoside bonding, while unexposed areas remain protected (Figure 1.3). The wafer is then flushed with a nucleoside-containing solution, and the nucleoside attaches at the activated sites. This process, which lengthens all activated probes by one nucleotide, alternates through the four nucleosides (A, C, G, and T) repeatedly to create 25-mers with the appropriate sequence in the appropriate spatial position on the wafer (Figure 1.4). Wafers are then diced and packaged individually into cartridges (Figure 1.5).

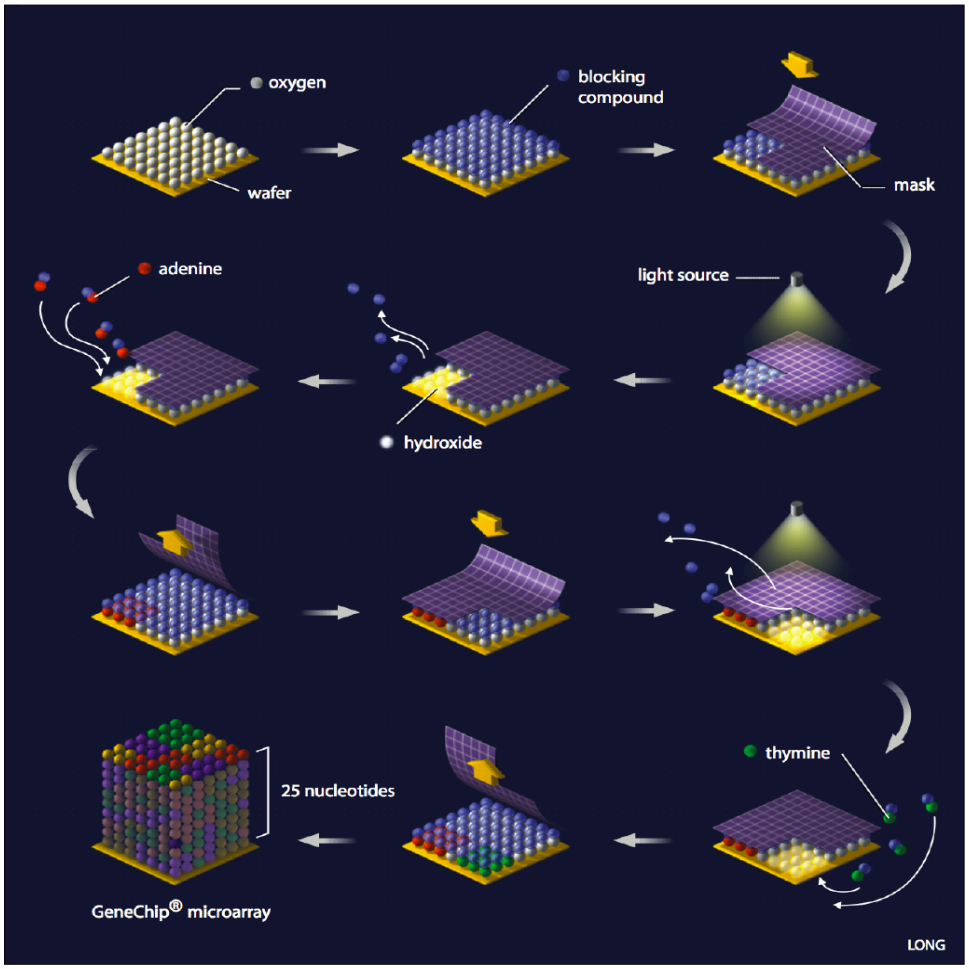

Figure 1.2: Overview of GeneChip synthesis. Nucleosides are progressively added to a silanized chip by photolithography to create a set of 25-mers matching specific sequences. Image courtesy of Science Creative Quarterly (http://scq.ubc.ca), Jiang Long, artist.

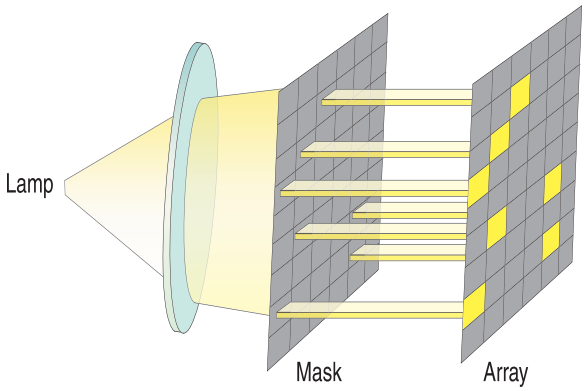

Figure 1.3: Activation of selected areas of a GeneChip using a photolithographic mask. Only probes in activated areas (shown in yellow) will be extended by the next application of nucleoside solution. Image courtesy of Affymetrix.

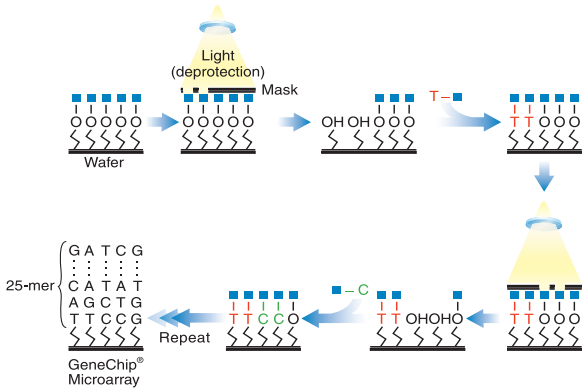

Figure 1.4: Synthesis of probes occurs in cycles, in which activated cells are lengthened by one nucleotide. Image courtesy of Affymetrix.

Figure 1.5: A completed GeneChip. Image courtesy of Affymetrix.

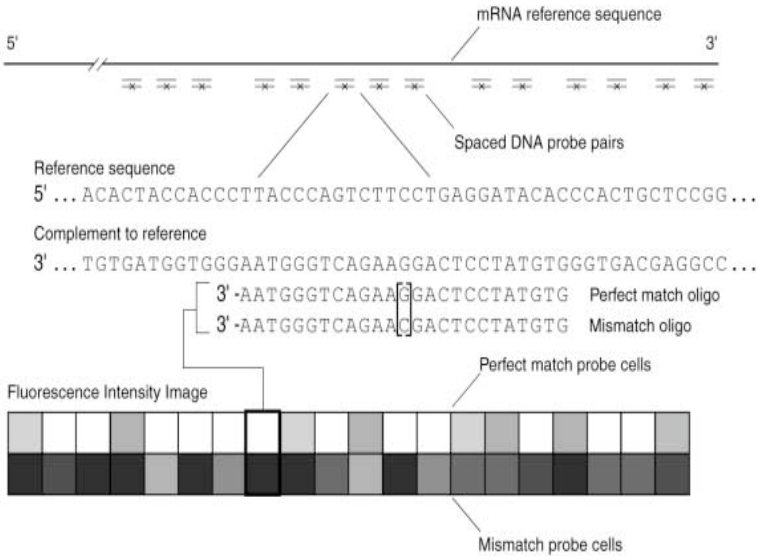

Figure 1.6: Structure of a GeneChip. Image courtesy of Affymetrix.

Probes are synthesized in sets of two called *probe pairs* (Figure 1.6). Each pair consists of a perfect match (PM) and a mismatch (MM) probe. The PM probe matches the target sequence exactly and is intended to measure specific signal for a particular gene. The MM probe has an inversion at the middle (13^{th}) base and is intended to measure nonspecific binding of the probe. Unfortunately, the PM probe may bind nonspecifically to other similar sequences (Wu and Irizarry, 2005), and the MM probe may have a specific binding component to the target sequence (Wu *et al.*, 2004); both effects have significant implications for the analysis of microarray data. A *probe set* consists of a group of 11 to 20 probe pairs, which are related by the fact that each pair interrogates a region on the same gene (Figure 1.6). This level of redundancy in the pairs of a probeset is intended to produce more accurate estimates of gene expression. There may be one or more probesets interrogating the same gene. For current chips, the majority of genes have only a single associated probeset. Finally, a GeneChip consists of thousands of probesets that interrogate multiple genes (Figure 1.6).

With the commercial Affymetrix GeneChip, one array is used for each sample in the experiment. The hybridization material for the arrays typically consists of mRNA isolated from the study samples, although DNA fragments have also been used. When mRNA is the starting material, a single strand of cDNA is produced through reverse transcription, followed by synthesis of

a second strand to produce double-stranded cDNA. Such steps are not necessary with DNA as a starting material. Next, an *in vitro* transcription (IVT) process is performed according to standard molecular biology procedures (Van Gelder *et al.*, 1990). This amplifies the target ribonucleic acid (RNA) for subsequent hybridizations. The use of biotinylated nucleosides in the IVT reaction allows subsequent detection of the target. The target RNA is then hybridized to the GeneChip and stained with a streptavidin-phycoerythrin conjugate using standard protocols available from Affymetrix (Affymetrix, 2002a). Following hybridization, the array is imaged using a fluorescent scanner. The fluorescent signal intensity is assumed to be proportional to the abundance of the target for each probe (Chudin *et al.*, 2002).

1.1.3 Comparison of cDNA and Oligonucleotide Array Technologies

Oligonucleotide and cDNA technologies represent two different approaches to the quantification of gene expression using microarrays. Recent studies have shown that the two are generally comparable in achieving this goal, and choice between platforms should be made based on other factors (Patterson *et al.*, 2006). Oligonucleotide chips require that the entire sequence of the target be known, while cDNA chips only require knowledge of the primer sequences for PCR (Hardiman, 2004). This can offer significant advantages in the study of less well characterized genes. The cDNA arrays are more easily customized (Hager, 2006), although made-to-order oligonucleotide chips are available (Dalma-Weiszhausz *et al.*, 2006). The cost per array is substantially lower for cDNA arrays, which may be produced using equipment available in most molecular biology laboratories (Hager, 2006). However, cDNA arrays may require dye swap experiments to separate dye effects from experimental effects, which may increase the number of arrays and offset any cost savings (Dobbin *et al.*, 2003). The quality of cDNA arrays that are not mass produced may be inferior to that of commercial oligonucleotide chips (Hardiman, 2004). Finally, the choice between cDNA and oligonucleotide arrays can have a significant influence on experimental design and analysis, as described in the next section.

1.2 Experimental Design

1.2.1 Introduction

Study design plays a critical role in obtaining meaningful results from microarray experiments (Reimers, 2005). It also has a significant influence on the selection and application of statistical tests that are used in the analysis. Traditional statistical topics such as power, randomization, and validation of model assumptions apply to microarray studies in a manner similar to biomedical studies, although these areas may receive insufficient attention in the planning of microarray experiments (Bolstad *et al.*, 2004). However, microarray experiments also have unique features not found in other types of studies, which are reviewed in this section. Detailed reviews of the topic for both cDNA and oligonucleotide arrays may be found in Bolstad *et al.* (2004), Churchill (2002), Kerr and Churchill (2001), and Simon *et al.* (2002).

One issue common to both cDNA and oligonucleotide arrays is replication (Allison *et al.*, 2006). Replication is essential for differentiating systematic variation due to treatment or phenotypic effects from chance variation (Lee, 2001). For microarray experiments, two types of replicates are possible, *technical* and *biological* (Yang and Speed, 2002). Technical replication occurs when RNA from the same extraction is hybridized to multiple chips. Biological replication occurs when RNA from different extractions of the same sample is hybridized to multiple chips, or when RNA from different samples within the same condition is hybridized to multiple chips. The variability associated with biological replicates is generally greater than that with technical replicates, as the variability between individuals (or extractions) is generally greater than variability in the hybridization and processing of chips. Technical replicates reduce variability, but only provide information about the variability within a particular extraction. In contrast, biological replicates provide information about the variability between individuals (or extractions), making the experimental results more generalizable. A combination of both technical and biological replicates is ideal, but biological replicates are preferred over technical replicates for both cDNA and microarray studies. Additional aspects of experimental design unique to each platform are

reviewed in the following sections.

1.2.2 Experimental Design of cDNA Microarray Studies

With the hybridization of two samples on a single array, cDNA experiments are inherently comparative (Yang and Speed, 2002). The results of a cDNA study measure the relative abundance of the target sequence in the experimental sample to that in the control sample. Since a control sample is hybridized to each chip, the arrangement of controls deserves special consideration (Churchill, 2002; Yang and Speed, 2002). For designs such as the comparison of gene expression pre- and post-treatment across several samples, the choice of control can be straightforward. However, with other types of experiments, such as the comparison of several experimental samples to a single control sample, the choice is less clear. More complicated designs, such as the common reference design and the loop design, frequently appear in cDNA studies. The latter designs are amenable to analysis with traditional statistical methods such as general linear models (GLMs), but introduce an added layer of complexity in the analysis.

Another consideration in the design of cDNA experiments is the effect of cohybridizing two samples to the same array simultaneously. This approach does have advantages, for it removes extraneous components of between-chip variability (such as differences in chip manufacturing or handling) when comparing the two samples on the chip (Duggan *et al.*, 1999). However, without appropriate care, it can also introduce unintended sources of confounding. Systematic differences between the red and green dyes can occur in the incorporation stage and in the scanning stage (Dobbin *et al.*, 2003). If not properly addressed, these dye biases can be mistaken for treatment effects. Dye-swap designs, in which the dyes for the experimental and control samples are swapped for different hybridizations, can be useful in overcoming this bias but at the expense of additional cost and complexity of analysis.

1.2.3 Experimental Design of Oligonucleotide Microarray Studies

In contrast to cDNA arrays, only one sample is hybridized to each chip for oligonucleotide arrays. This can greatly simplify experimental design, making the process very similar to experimental design for biomedical studies, which has been extensively investigated (Bolstad *et al.*, 2004). Thus, variations of the GLM are the most frequently used approach (Jafari and Azuaje, 2006). However, the use of one sample per chip may potentially confound differences in chip manufacturing and processing with treatment differences (Gebicke-Haerter, 2005). Replication can help prevent this bias, as can appropriate use of randomization in the processing phase, although the latter may not be adequately considered in many microarray experiments. Preprocessing procedures such as normalization are intended to reduce this bias once it occurs, although this only achieves a partial correction (Bolstad *et al.*, 2004).

1.2.4 Notational Conventions

Notation used in the experimental design of microarray studies varies considerably by author. In all subsequent sections, probes (or probe pairs) will be designated using the variable P and the subscript p, probesets with the variable S and subscript s, arrays or chips with the variable M and subscript m, and experimental conditions or classes with the variable C and subscript c. In cDNA array models, dye will be designated using the variable D and subscript d and gene, which roughly corresponds to the probeset variable for oligonucleotide arrays, will be designated with the variable G and subscript g. For those experiments having both cell lines and treatments, the former will be denoted with the variable L and subscript ℓ, and the latter with the subscript for condition. The variable N with the appropriate subscript will be used to denote the number of elements at that level. For example, N_p will denote the number of probes within a probe set, which may vary from probeset to probeset. Scalar quantities will be denoted with italicized Roman or Greek letters, which may be upper- or lowercase, and the upper- and lowercase variables are distinct. Vector quantities will be denoted with bold lowercase and matrix quantities with bold uppercase Roman or Greek letters. This may lead to slight notational differences from the

original, cited articles, although the meaning will remain the same. Abbreviations for the classifications of differential gene expression produced by software, such as P for present and A for absent, are not intended to represent variables and will not be italicized.

1.3 Image Analysis

Following hybridization, both cDNA and oligonucleotide arrays are scanned using a fluorescent scanner, producing an image of the chip with areas of higher intensity corresponding to areas of greater target concentration. This image is stored as a graphics file, typically using the tagged image file format (TIFF) structure. The process of image analysis converts this graphics file into a series of numerical values, one for each probe, which represents the signal intensity for that probe. This is a complex process involving three steps: *gridding*, which assigns coordinates to the pixels on the chip; *segmentation*, in which each pixel in the image grid is classified as background or signal, and *intensity extraction* or *quantification*, in which a single overall estimate for the signal intensity for the probe is computed from the set of pixels representing signal for that probe (Yang *et al.*, 2001). Image analysis produces a file of signal intensities, whose exact format varies by platform, which are suitable for expression analyses as detailed in the following sections. Algorithm development for image analysis is an active area of research; an overview of the topic for cDNA arrays is given by Jain *et al.* (2002) and Yang *et al.* (2001) and for oligonucleotide arrays by Schadt *et al.* (2001).

For Affymetrix GeneChips, image analysis produces a CEL file, which contains information about the experiment and the probe intensities. The first component of the CEL file is a header section, which details the physical characteristics of the chip and the parameters for the image analysis. This is followed by an intensity section, which gives the physical x and y coordinates of the probe on the chip, the number of pixels assigned to the probe, and the mean and standard deviation for the probe based on the pixel intensities. Next is a section giving the coordinates of probes that are outliers. The final two sections identify masked probes that are removed from analysis by the user and modified probes with intensities that are assigned by the user; these are listed in the

final two sections. Using this format, the CEL file provides all of the data necessary for analysis of individual microarrays. Further details of the CEL file format are available from the Affymetrix website at http://www.affymetrix.com/Auth/support/developer/fusion/file_formats.zip.

1.4 Early Methods for Computing GeneChip Expression

One of the first widely used methods for analysis of GeneChip data was the Affymetrix Microarray Suite version 4.0 (MAS4) algorithm, also called the Empirical Expression algorithm (Affymetrix, 2004). As the CEL file format was originally proprietary, MAS4 was the *de facto* standard for expression analysis of Affymetrix data in early studies. It also introduced the concepts of background correction and summarization for GeneChip data, which have been incorporated in the preprocessing step of later algorithms for Affymetrix expression analysis.

The full details of the MAS4 algorithm have not been published, but a limited description is provided in the Affymetrix MAS4 User's Guide and the GeneChip Operating Software (GCOS) manual (Affymetrix, 2004). For single array analyses (or for each chip in a multi-chip comparison), the MAS4 algorithm generates three different metrics comparing the PM and MM intensities of each probe pair in a probe set. These metrics are then weighted and entered into a decision matrix to determine the *absolute call*, which represents the overall status of the probeset on the chip. The absolute call is simply a classification of present (P), marginally present (M), or absent (A) for each probeset. In addition, the MAS4 algorithm calculates the $AvgDiff$ value for each probeset, which is a relative measure of the level of expression for the probeset. The $AvgDiff$ for the sth probeset is defined as

$$AvgDiff_s = \frac{1}{N_p} \sum_{p=1}^{N_p} \left(PM_{sp} - MM_{sp} \right) \tag{1.1}$$

where N_p is the number of probe pairs in probeset s after trimming values with extremely strong or extremely weak intensity values. Background correction is performed by subtracting the MM signal, which is intended to measure nonspecific binding for a probe, from the PM signal to

obtain an estimate of specific binding. The multiple probe-level values are then summarized into a single expression index for the probeset using the arithmetic mean.

Comparisons between arrays in the MAS4 algorithm are made by comparing each experimental array in turn to a user-selected baseline array. The algorithm generates five different metrics for each comparison. These metrics are then weighted and entered into a decision matrix to determine the *difference call*, which represents the status of the probeset on the experimental array relative to the baseline array. The difference call classifies the changes in expression for each probeset as increased (I), mildly increased (MI), no change (NC), mildly decreased (MD), or decreased (D). No original validation studies of the MAS4 algorithm were published, although it has been used in peer-reviewed publications.

1.5 Current Methods for Computing GeneChip Expression

1.5.1 Introduction

The development of MAS4 represents a significant step forward in the analysis of microarray data, as it provided a standard index of expression that corresponds to concentration in many circumstances (Hubbell *et al.*, 2002). However, investigators quickly realized some of the limitations of the MAS4 algorithm. The estimates are sensitive to outliers (Hubbell *et al.*, 2002). Given the large number of probes on a GeneChip, the presence of outliers is quite likely and can dramatically influence results. The use of the raw MM probe values to estimate nonspecific binding is also problematic, as the MM value exceeds the PM value for about a third of the probes (Irizarry *et al.*, 2003). This corresponds to a negative estimate of the target concentration, a physical impossibility. Furthermore, the negative estimates preclude the use of many transformations, such as the logarithmic, that are commonly used in statistics.

With the release of the Affymetrix file formats into the public domain (available at http://www.affymetrix.com/Auth/support/developer/fusion/file_formats.zip), many researchers began to develop their own analytical algorithms to correct the deficiencies of MAS4. In the following

Method	Background Correction	Normalization	Summarization
MAS4	Mismatch intensity subtraction	None	Arithmetic Mean
MAS5	Idealized Mismatch intensity subtraction	Scale	Tukey's Biweight
dChip	Model-Based Expression Index	Invariant Set	Model-Based Expression Index
RMA	Model-based background subtraction	Quantile	Median Polish

Table 1.1: Overview of Probeset Summary Methods for Affymetrix Data.

sections, the shared features of these algorithms will be presented, followed by specific details of some of the more commonly used ones. These examples will illustrate the variety of approaches currently used in microarray data analysis and serve as a reference for methods used in later comparative studies. An overview of these algorithms is presented in Table 1.1.

1.5.2 Pre-processing

Most algorithms for Affymetrix data that are in current use perform pre-processing of the microarray data. Pre-processing is intended to correct potential problems when obtaining intensity values for the probesets on a GeneChip, thus preparing the data from image analysis algorithms for statistical analysis to detect differential gene expression (Bolstad *et al.*, 2005). Pre-processing consists of three steps: background correction, normalization, and summarization. Background correction removes the signal resulting from nonspecific binding of target to probe. Specific binding occurs due to complementary base-pairing between the two sequences, and the goal of gene expression studies is to measure specific binding precisely (Wu and Irizarry, 2005). Nonspecific binding occurs due to other factors, spuriously increasing the detected signal. The MM probe in a probe pair is intended to measure this nonspecific binding, so that the $PM - MM$ difference would be specific binding (Wu and Irizarry, 2005). However, the relationship has proven to be more complex, as the MM probe measures some specific as well as nonspecific binding (Wu

et al., 2004). Because of this, some investigators have chosen only to use PM probes for estimation of background, while others estimate the background with models utilizing both PM and MM values (Li and Wong, 2001).

Normalization adjusts the probe intensities to make measurements between different chips comparable (Bolstad *et al.*, 2003). Variations in the intensity measurements between samples are expected to occur due to differential gene expression. However, another level of variation occurs for reasons other than biological differences, which Bolstad *et al.* call *obscuring variation*; these sources of variation can include technical differences in sample preparation, hybridization, and scanning (Zakharkin *et al.*, 2005). This additional variation leads to differences in the distribution of the probe intensities among chips, making direct comparisons difficult. Normalization manipulates the probe intensity distributions in an attempt to minimize obscuring variation, so that valid statistical comparisons of the remaining biological variation can be performed.

Summarization combines the information from the many probes targeting a gene into a single number representing the expression level for that gene. From a statistical standpoint, this is a data reduction, which can be useful for reducing the time and complexity of subsequent calculations. The results of summarization also correspond to the biological question being posed, as expression differences between genes, rather than expression differences between probes, are usually of interest.

1.5.3 dChip

One of the first widely-used alternatives to MAS4 was the algorithm used in the dChip software (Li and Wong, 2001; Li and Hung Wong, 2001). For normalization, dChip uses the invariant set method, which attempts to base normalization only on those probes that are not differentially expressed between chips (Li and Hung Wong, 2001). Such probes would be expected to have similar (but not necessarily identical) intensity-based ranks between two chips, with one chip identified as the baseline and the other as the chip to be normalized. The procedure to identify these probes, called the invariant set, is detailed in Algorithm 1.1.

Algorithm 1.1 Identification of Invariant Set

1: Initialize the invariant set at iteration zero $IS^{(0)}$ to the set of all PM probes for the two chips.
2: For iteration i, calculate the proportional rank difference (PRD), defined as the rank difference between the two chips divided by the number of probes per chip, for each of the PM probes in the dataset $IS^{(i)}$.
3: If the PRD for a probe is less than the threshold δ, defined as $\delta = 0.003$ for low intensity probes and $\delta < 0.007$ for high intensity probes, then retain the probe for the dataset $IS^{(i+1)}$ of the next iteration. (The different thresholds allow for the sparsity of probes at the upper tail of the intensity distribution.)
4: Repeat steps 2 and 3 until the invariant set does not change between iterations.
5: Fit a piecewise linear running median line using the invariant set and perform normalization by projecting the intensities of the chip to be normalized onto the line.

For background correction and summarization, dChip calculates a model-based expression index (MBEI). This uses a statistical model for the probe intensities that assumes the intensity is directly proportional to the expression index (or microarray effect) M for array m, but that the proportionality constant may differ by probe. Furthermore, the increase in PM intensity will always be greater than the increase in MM intensity. The model is

$$MM_{mp} = \mu_p + M_m\phi_p + \varepsilon_{mp} \tag{1.2}$$

$$PM_{mp} = \mu_p + M_m\phi_p + M_mP_p + \varepsilon_{mp} \tag{1.3}$$

where μ is the baseline intensity for the pth probe due to nonspecific binding, ϕ is the proportionality constant for the MM probe due to specific binding, P is the proportionality constant for the PM probe due to specific binding, and the error term $\varepsilon \sim N\left(0, \sigma^2\right)$. For identifiability, the constraint $\sum_{p=1}^{N_p} P_p = N_p$ is also imposed.

In the original model (Li and Wong, 2001), the $PM - MM$ differences are used as the estimate of the specific binding for the probe

$$y_{mp} = PM_{mp} - MM_{mp} = M_mP_p + \varepsilon_{mp} \tag{1.4}$$

Estimates of the parameters are found using Equation (1.4). An iterative least squares fitting is performed for M and for P, with one parameter being fit while holding the other parameter

constant. The interpretation of the MM signal intensity can be problematic, so Li and Hung Wong (2001) later constructed a PM-only model

$$PM_{mp} = \mu_p + M_m P_p + \varepsilon_{mp}$$

This model is fit using iterative least squares, similar to the $PM - MM$ model. Unfortunately, the use of the dChip algorithm requires a rather large number of arrays to obtain accurate estimates; Li and Wong recommend 10 or more for most experiments. The software is available at the dChip website (http://www.dchip.org). The algorithm is also implemented in the *fit.li.wong* and *expresso* functions in the R package *affy*. However, since the dChip software is not open source, the latter two functions do not exactly reproduce the results of the former.

1.5.4 MAS5

The MAS5 algorithm was developed by Affymetrix as a successor to MAS4 (Liu *et al.*, 2002; Hubbell *et al.*, 2002). MAS5 made significant changes to the background correction, normalization, and summarization procedures to overcome deficiencies noted in MAS4.

MAS5 makes extensive use of the one-step Tukey's biweight procedure, which is a robust measure of central tendency somewhat similar to a weighted mean (Hoaglin *et al.*, 2000). It is based on a uniform distance measure is calculated for each observation, with standardization using the median and median absolute deviation (MAD). The biweight value then calculated as a weighted average of each of the observations. The weight function is chosen to give a weight that decreases with distance for observations within a certain distance from the median, and a zero weight for observations outside of this distance from the median. Tukey's biweight $T_{bi}(\cdot)$ for the n observations $x_1, x_2, \ldots, x_n$ is formally defined as

$$T_{bi}(x_1, x_2, \ldots, x_n) = \frac{\sum_{i=1}^{n} w_{bi}(u_i)\, x_i}{\sum_{i=1}^{n} w_{bi}(u_i)}$$

where $w_{bi}(\cdot)$ is the weight function

$$w_{bi}(x_i) = \begin{cases} \left(1 - [u_i(x_i)]^2\right)^2, & |u_i(x_i)| \leq 1 \\ 0, & |u_i(x_i)| > 1 \end{cases}$$

and $u(\cdot)$ is the uniform distance measure

$$u_i(x_i) = \frac{x_i - \text{median}(x_1, \ldots, x_n)}{\tau_1 \cdot \text{MAD}(x_1, \ldots, x_n) + \tau_2}$$

Both τ_1 and τ_2 are user-defined tuning constants specified in the MAS5 software. Tukey's biweight can be computed by iteratively reweighting the estimates until convergence occurs, but the one-step estimation used by MAS5 will generally produce acceptable results (Hoaglin *et al.*, 2000).

For background correction, MAS5 uses a multistep process to ensure that negative intensity estimates do not occur. Each chip is divided into an $N_z \times N_z$ set of rectangular regions, called *zones*. The zone background bZ is the average of the lowest 2% of probe intensities for each zone. The background bg for a probe at position (x, y) on the chip is calculated as a weighted sum of the zone background values, which smoothes the transition between zones:

$$bg = \frac{1}{\sum_{i=1}^{N_z^2} w_i(x, y)} \sum_{i=1}^{N_z^2} w_i(x, y)\, bZ_i \tag{1.5}$$

where $w_i(\cdot)$ is the weight function

$$w_i(x, y) = \frac{1}{d_i^2(x, y) + \tau_{sm}} \tag{1.6}$$

and $d(\cdot, \cdot)$ is a distance function, the form of which is not specified by Affymetrix. Here the (x, y) coordinates of a zone are defined to be the center of the zone, and τ_{sm} is a user-defined tuning constant for adjusting the degree of smoothing. The background value for each probe subtracted from the PM and MM intensities for that probe; any negative values are replaced by a user-defined

fraction of the noise for that probe. Next, a probeset-level specific background SB for probeset s is calculated as

$$SB_s = T_{bi}\left(\log_2 PM_{s1} - \log_2 MM_{s1}, \ldots, \log_2 PM_{s,N_p} - \log_2 MM_{s,N_p}\right)$$

Background correction concludes with the calculation of the idealized mismatch IM. The $PM - IM$ value is intended to measure specific binding, but unlike the $PM - MM$ value is guaranteed never to be negative. The formula for the IM value is

$$IM_{sp} = \begin{cases} MM_{sp}, & MM_{sp} < PM_{sp} \\ \dfrac{PM_{sp}}{2^{SB_s}}, & MM_{sp} \geq PM_{sp} \text{ and } SB_s > \tau_c \\ \dfrac{PM_{sp}}{2^{\left(\frac{\tau_c}{1+\frac{\tau_c - SB_s}{\tau_s}}\right)}}, & MM_{sp} \geq PM_{sp} \text{ and } SB_s \leq \tau_c \end{cases}$$

where τ_c and τ_s are user-defined tuning constants (contrast and scale, respectively) for weighting the relative importance of probe- and probeset-level information in the calculation. Using this formula, if the MM value is already lower than the PM value, then MM is a reasonable estimate of nonspecific binding and is used for the IM value. Otherwise, the IM value is calculated as a fraction of the PM value. If specific background SB is "large" (as defined using the τ_c parameter), then the intensities from the probeset are generally reliable, and the IM value that is constructed is not specific to the individual probe but does use information restricted to the associated probeset. If SB is "small", the intensities from the probeset are less reliable, and the IM value that is constructed is only weakly related to the probe- and probeset-specific information.

Summarization in MAS5 creates a single expression log value LV for each probeset by taking the Tukey's biweight of the log difference between the PM and IM values:

$$LV_s = T_{bi}\left(\log_2\left(PM_{s1} - IM_{s1}\right), \ldots, \log_2\left(PM_{s,N_p} - IM_{s,N_p}\right)\right)$$

Normalization in MAS5 is performed by conducting a scale normalization, which adjusts all chips to have the same mean intensity. This is done by calculating a scale factor SF to adjust the 2% trimmed mean to a user-defined target intensity value, then multiplying the unadjusted expression values by the scale factor to obtain the final expression values EV.

$$SF = \frac{tg}{\text{TrimMean}\left(2^{LV_s}, 0.02, 0.98\right)}$$

$$EV_s = SF \cdot 2^{LV_s}$$

MAS5 is implemented in the GCOS software available from Affymetrix. It is also implemented in the *expresso* and *mas5* functions in the R package *affy*, available from the Bioconductor project (http://www.bioconductor.org). However, the latter two functions do not exactly reproduce the results of the Affymetrix software, as they were developed prior to the release of the GCOS software as open source (see http://www.affymetrix.com/support/developer/stat_sdk/index.affx).

1.5.5 RMA

The robust multi-chip average, or RMA (Irizarry *et al.*, 2003), is one of the most widely used expression measures and a standard choice for comparing the performance of new algorithms. RMA utilizes only the PM value in calculations. For background correction, the observed intensities are modeled as a combination of probe-specific signal sg and nonspecific background bg:

$$PM_{msp} = sg_{msp} + bg_{msp}$$

where sg has an exponential distribution, $bg \sim N\left(0, \sigma^2\right)$, and sg and bg are independent of each other. Background adjustment is accomplished by estimating the probe-specific intensities $\mathcal{E}\left(sg_{msp} | PM_{msp}\right)$, which are the quantities of interest.

For normalization, RMA uses the quantile normalization method (Bolstad *et al.*, 2003). It is an aggressive form of normalization that forces each chip to have the same distribution of intensities. This is accomplished by transforming all chips to a common distribution, standardizing, and then back-transforming each chip individually, as shown in Algorithm 1.2. The elements of

Algorithm 1.2 Quantile Normalization

1: Arrange the m microarrays, each with p probesets, into a $m \times p$ matrix $\boldsymbol{Y}$
2: Sort each column of $\boldsymbol{Y}$ individually to give $\boldsymbol{Y}_{sort}$
3: Compute the row means of $\boldsymbol{Y}_{sort}$
4: Assign the row mean to each element of the row to give $\boldsymbol{Y}_{adj}$
5: Rearrange each column of $\boldsymbol{Y}_{adj}$ into the original unsorted order in $\boldsymbol{Y}$ to give $\boldsymbol{Y}_{norm}$

$\boldsymbol{Y}_{norm}$, denoted by y, represent the quantile-normalized intensities that are used in subsequent computations. Because quantile normalization forces the same distribution across chips, the authors recommend that normalization only be performed on similar chips.

For computing expression summary values, RMA fits the background-adjusted, quantile-normalized, and $\log_2$-transformed PM intensities y to the additive model

$$y_{msp} = \mu_s + P_{sp} + M_{ms} + \varepsilon_{msp}$$

where μ denotes the overall mean for the probeset, P represents the probe affinity effect, and M represents the array effect, and the error term $\varepsilon \sim N\left(0, \sigma^2\right)$. The estimate $\hat{\mu} + \hat{M}_{ms}$ is the probeset expression summary for probeset s on array m. This model is fit using an iterative median polish procedure, which is a robust method of fitting an additive model that is in some ways analogous to analysis of variance (ANOVA) (Hoaglin *et al.*, 2000, chapter 6). This procedure iteratively subtracts the median across each factor (median polish) to obtain successive approximations for the factor effects, as shown in Algorithm 1.3. Iterations may be carried out until convergence, which occurs when the residuals after the median polish are zero across all factors, but a small number of iterations are generally sufficient to give good estimates of the factor effects. The RMA algorithm is available in the *rma* and *expresso* functions in the R package *affy*, as well as the stand-alone RMAExpress software (http://rmaexpress.bmbolstad.com/).

Algorithm 1.3 Median Polish

1: Initialize the residuals to the intensity data.
$\varepsilon_{msp}^{(0)} \leftarrow y_{msp}$
2: Initialize the row (probe), column (array), and overall (mean) effects to 0.
$\mu_s^{(0)} \leftarrow 0$
$P_{sp}^{(0)} \leftarrow 0$
$M_{ms}^{(0)} \leftarrow 0$
3: Row polish by subtracting row (probe) medians
$\Delta P_{sp}^{(i)} \leftarrow \text{median}\left(\varepsilon_{1sp}^{(i-1)}, \ldots, \varepsilon_{N_m,sp}^{(i-1)}\right), p = 1, \ldots, N_p$
$\Delta\mu_{Ms}^{(i)} \leftarrow \text{median}\left(M_{1s}^{(i-1)}, \ldots, M_{N_m,s}^{(i-1)}\right)$
$\varepsilon_{msp}^{(i)} \leftarrow \varepsilon_{msp}^{(i-1)} - \Delta P_{sp}^{(i)}, m = 1, \ldots, N_m; p = 1, \ldots, N_p$
4: Column polish by subtracting column (array) medians
$\Delta M_{ms}^{(i)} \leftarrow \text{median}\left(\varepsilon_{ms1}^{(i)}, \ldots, \varepsilon_{ms,N_p}^{(i)}\right), m = 1, \ldots, N_m$
$\Delta\mu_{sP}^{(i)} \leftarrow \text{median}\left(P_{s1}^{(i-1)} + \Delta P_{s1}^{(i)}, \ldots, P_{s,N_p}^{(i-1)} + \Delta P_{s,N_p}^{(i)}\right)$
$\varepsilon_{msp}^{(i)} \leftarrow \varepsilon_{msp}^{(i)} - \Delta M_{ms}^{(i)}, p = 1, \ldots, N_p; m = 1, \ldots, N_m$
5: Estimate the effects
$\mu_s^{(i)} \leftarrow \mu_s^{(i-1)} + \Delta\mu_{sP}^{(i)} + \Delta\mu_{Ms}^{(i)}$
$P_{sp}^{(i)} \leftarrow P_{sp}^{(i-1)} + \Delta P_{sp}^{(i)} - \Delta\mu_{sP}^{(i)}, p = 1, \ldots, N_p$
$M_{ms}^{(i)} \leftarrow M_{ms}^{(i-1)} + \Delta\mu_{Ms}^{(i)} - \Delta\mu_{Ms}^{(i)}, m = 1, \ldots, N_m$

1.6 Hypothesis Testing

Preprocessing of Affymetrix data generates expression summary values that quantify expression levels of genes on each chip, but expression summary values do not provide information regarding differential expression of genes between conditions. The earliest and simplest approach for determining differential expression was to examine the fold change, which is the ratio of the expression summary values for the two conditions of interest. However, investigators quickly realized that fold change was an inadequate measure, as it does not account for the variability of expression measurements (Yang *et al.*, 2002). Statistical tests of hypotheses are necessary to assess the reliability of findings from microarray experiments (Firestein and Pisetsky, 2002). Many different types of analyses are suitable for microarray data, but variants of the GLM are among the most widely used, particularly for assessing differential expression (Jafari and Azuaje, 2006). Although it is possible to combine preprocessing with hypothesis testing in the analytical workflow, the two are usually separate (Tumor Analysis Best Practices Working Group, 2004).

The use of GLM methodology brings an extensive literature to bear on microarray data analysis, including such issues as design of experiments, sample size determination, and testing for specific types of designs. Certain aspects of statistical hypothesis testing assume particular importance in microarray studies. The first is how to appropriately estimate the variance used for the hypothesis tests (Cui and Churchill, 2003). One approach is to model the variances as being completely homogenous, so that a single common variance is used for testing the significance of all genes. Such an approach offers much greater power than other methods. However, it is generally not biologically plausible, as the amount of variation between genes is usually considerable (Wright and Simon, 2003). Furthermore, when using a single global variance, rankings based on the p-values of the t or F tests under GLM reduce to rankings based on the fold change (Cui and Churchill, 2003), which has already been noted to be problematic. Another approach is to model the variances as being completely heterogenous, with a separate variance for each gene. This is much more reasonable biologically, but suffers from low power to detect significant differences that typically makes it unsuitable for analyses. Thus, the most common approach is to combine the two into a "moderated" variance estimate. This allows the variance to differ by gene, but borrows information across genes to obtain a more reliable variance estimate.

A number of approaches have been proposed for borrowing information across genes. A Bayesian approach imposes a prior distribution on the variances, with estimates for the variances being obtained by Bayes or empirical Bayes procedures (Baldi and Long, 2001; Lonnstedt and Speed, 2002; Kendziorski *et al.*, 2003; Newton *et al.*, 2001; Efron *et al.*, 2001; Smyth, 2004). This is implemented in the *limma* package available from Bioconductor (Smyth, 2005). Frequentist methods for moderating the variance have also been developed. The t-test in the Significance Analysis of Microarrays (SAM) software (http://www-stat.stanford.edu/~tibs/SAM/) adds a small bias constant to the gene-specifc variances to attenuate the effect of small variances (Tusher *et al.*, 2001). Although this does moderate the variances, it fails to utilize information across genes. Other researchers suggest that the gene-specific variances be considered as a sample from a common distribution, which shares similarities to the Bayesian approach but leads to different estimators (Wright and Simon, 2003). Finally, Cui *et al.* (2005) describe a shrinkage estimator

		Decision		
		Accept	Reject	Total
H_0	True	n_1	n_2	H_T
	False	n_3	n_4	H_F
	Total	A	R	N

Table 1.2: Outcomes from N Simultaneous Hypothesis Tests, where $n_1 + n_4$ is the total number of correct decisions, n_2 is the number of type I errors, and n_3 is the number of type II errors.

for the gene-specific variances based on the James-Stein estimator.

A second aspect of particular importance in microarray studies is how to adjust for multiple hypothesis testing. Microarray experiments measure the expression of thousands of genes at once, yet the number of arrays hybridized is typically small. This is the inverse of the usual statistical analysis, where only a small number of hypotheses are examined, and numerous replications are used to achieve accuracy. Thus, traditional methods of adjusting for multiple testing may not be suitable for microarray data. Assuming that N simultaneous hypothesis tests are being conducted with an desired overall type I error rate of α, the outcomes may be conceptualized as shown in Table 1.2. Traditionally, adjustment for multiple hypothesis testing has been accomplished by controlling the family-wise error rate (FWER; Dudoit *et al.*, 2002). This rate is computed as

$$FWER = Pr\,(n_2 \geq 1) \tag{1.7}$$

Thus, the family-wise error rate (FWER) is the probability of one or more type I errors (or false positives) among all genes declared differentially expressed. The FWER is controlled by conducting the N individual tests at lower significance level so that the overall level remains at α. The prototypical example is the Bonferroni procedure, which adjusts the level of the individual tests to be α/N. Less conservative methods of adjusting the FWER have also been developed.

Benjamini and Hochberg (1995) proposed an alternative to the FWER, which they call the *false discovery rate* (FDR). The false discovery rate (FDR) represents the expected proportion of false positives among all of the rejected null hypotheses (false discoveries) times the probability

of at least one rejected null hypothesis. This is computed as

$$FDR = \mathcal{E}\left(\frac{n_2}{R} \mid R > 0\right) \times Pr\,(R > 0) \tag{1.8}$$

The FDR is generally a more meaningful method of controlling error rates for microarray studies, which are exploratory in nature and so a small number of false positives is acceptable. It also has the practical interpretation that 100α percent of the genes declared differentially expressed would be false positives. The FDR controlling procedure may be expressed algorithmically as follows: Let N be the number of null hypotheses, with H_T of these being true and H_F of these being false. Let $H_1, H_2, \ldots, H_N$ represent the null hypotheses corresponding to the vector of test statistics $T_1, T_2, \ldots, T_N$. Denote the associated p-values as $p_1, p_2, \ldots, p_N$ so that $p_i = 1 - F_{H_i}\,(T_i)$ for the cumulative distribution function (CDF) $F\,(\cdot)$ under the null hypothesis H_i. Let $p_{(1)}, p_{(2)}, \ldots, p_{(N)}$ be the ordered p-values, and let $H_{(i)}$ be the hypothesis corresponding to $p_{(i)}$. Define

$$j = \max\left\{i : p_{(i)} \leq \frac{i}{N}q\right\} \tag{1.9}$$

Then, for independent test statistics and for any configuration of false null hypotheses, rejecting all $H_{(i)}, i = 1, 2, \ldots, j$, controls the FDR at level q.

The above procedure (hereafter called the Benjamini-Hochberg procedure) assumes that the N null hypotheses are independent, which would seem to be an untenable assumption for microarray studies (Dudoit *et al.*, 2002). However, the proof of the associated theorem does not require independence, and it is possible that this assumption may be relaxed. Benjamini and Yekutieli (2001) showed that the above result for the Benjamini-Hochberg procedure holds under more general conditions for multivariate normal test statistics. Reiner *et al.* (2003) note that these results for FDR control with one-sided tests apply to microarray experiments; assuming that up- and down-regulation are equally likely to occur, these results extend to two-sided tests as well (Yekutieli, 2002). Benjamini and Yekutieli (2001) further generalize the FDR controlling procedure to allow for arbitrary dependency structures, but this result comes with a considerable

loss of power compared to the Benjamini-Hochberg procedure and is not necessary for most studies (Reiner *et al.*, 2003).

1.7 Mixed Effects Models for Microarrays

1.7.1 Introduction

Kerr, Martin, and Churchill (2000) were the first to propose the use of ANOVA models to adjust for the multiple sources of variation in microarray data. Their original formulation, which is for cDNA microarrays rather than Affymetrix GeneChips, models the log-transformed intensity y of the m^{th} microarray, d^{th} dye, c^{th} condition (labeled as *variety* in the original paper), and g^{th} gene as

$$y_{cmdg} = \mu + M_m + D_d + C_c + G_g + (MG)_{mg} + (CG)_{cg} + \varepsilon_{cmdg} \tag{1.10}$$

where μ is the overall mean, M is the array effect, D is the dye effect, C is the effect of condition, G is the gene effect, (MG) is the array by gene interaction, (CG) is the condition by gene interaction. In the vector and matrix formulations of the model described in subsequent sections, these effects would be incorporated into the design vector (or matrix), denoted $\boldsymbol{x}$ (or $\boldsymbol{X}$), and the vector (or matrix) of coefficients, denoted $\boldsymbol{\beta}$. The error terms ε are assumed to be iid $N\left(0, \sigma^2\right)$. For this model, the effects of interest are the condition by gene interactions (CG). A nonzero value of this term would be interpreted as differences in intensity across conditions for gene g, indicating differential expression. Although replicates were not explicitly incorporated into the description by Kerr and colleagues, the flexibility of the ANOVA model allows such information to be added easily.

Wolfinger *et al.* (2001) extended this model for cDNA arrays to include random effects. They divide the analysis into two interconnected ANOVA models. The first is a "normalization" model corresponds to the normalization step in expression summary measures, correcting for systematic differences among arrays to allow valid comparisons. For this model, the log-transformed intensity y of the c^{th} condition (labeled as *treatment* in the original paper), m^{th} microarray, and g^{th} gene

may be written as

$$y_{cmg} = \mu + C_c + M_m + (CM)_{cm} + \varepsilon_{cmg} \tag{1.11}$$

where μ is the overall mean, C is the condition effect, M is the array effect, (CM) is the array by condition interaction, and ε represents random error. The effects M, (CM), and ε are assumed to be normally distributed random variables with mean 0 and variances of σ^2_M, σ^2_{CM}, and σ^2_ε respectively. These random variables are also assumed to be independent of each other. Wolfinger *et al.* did not include dye effects in Equation (1.11) as dye effects were confounded with treatment effects in their experiment. However, dye effects may be added to the normalization model when permitted by the experimental design. The residuals $\hat{\varepsilon}$ from Equation (1.11) contain the remaining variation after adjusting for treatment and array effects. These residuals are used as the dependent variables in the second "gene" model, which is given by

$$\hat{\varepsilon}_{cmg} = G_g + (CG)_{cg} + (MG)_{mg} + \psi_{cmg} \tag{1.12}$$

where G is the gene effect, (CG) is the gene by condition interaction, (MG) is the gene by array interaction, and ψ is random error. The effects (MG) and ψ are assumed to be normally distributed random variables with mean 0 and variances $\sigma^2_{MG_g}$ and $\sigma^2_{\psi_g}$ respectively. Again, the random variables are assumed to be independent of each other. The availability of replicate arrays is explicitly assumed as significance levels are determined using estimates of within-gene variability. While the model in Equation (1.10) treats arrays as a fixed effect, Equations (1.11) and (1.12) model arrays as a random sample from a population of arrays. Similar to Equation (1.10), the effects of interest are the gene by treatment interactions.

The ANOVA models described above fit a single model to all genes on the array. Ayroles and Gibson (2006) note that, with larger data sets, the error variance may be estimated within genes. They propose a gene-specific ANOVA in which a linear model is fit separately for each gene. The relative fluorescence intensities y for the m^{th} microarray, d^{th} dye, and c^{th} condition (or treatment

in the original paper) would be

$$y_{cmd} = \mu + M_m + D_d + (MD)_{md} + C_c + \varepsilon_{cmd} \tag{1.13}$$

where μ is the overall mean, M is the array effect, D is the dye effect, (MD) is the array by dye interaction, C is the condition effect, and ε represents random error. The error term ε is assumed to be normally distributed with mean 0 and variance σ^2, and may be different for each gene. The gene-specific ANOVA model more accurately captures gene-to-gene variation present on an array, which is a definite advantage for statistical analysis when there are sufficient replicates for such an approach.

Chu *et al.* (2002) modified the mixed effects model proposed by Wolfinger *et al.* for the analysis of Affymetrix GeneChip data. In this formulation, the dependent variable is a suitable measure of gene expression for the Affymetrix platform. Chu *et al.* recommend the $\log_2$-transformed PM intensities, but the $\log_2(PM - MM)$ values may also be used after appropriate adjustment for negative values. They further recommend that the dependent variable be centered to have mean 0 to adjust for gross array-level effects. The transformed and centered intensities y_p for the ℓ^{th} cell line, c^{th} condition (or treatment in the original paper), p^{th} probe, and m^{th} array are modeled as

$$y_{\ell cmp} = L_\ell + C_c + (LC)_{\ell c} + P_p + (LP)_{\ell p} + (CP)_{cp} + M_{m(\ell c)} + \varepsilon_{\ell cmp} \tag{1.14}$$

where L is the cell line effect, C is the condition (treatment in the original paper) effect, (LC) is the cell line by condition interaction, P is the probe effect, (LP) is the cell line by probe interaction, (CP) is the condition by probe interaction, M is the array effect nested within cell line and condition, and ε represents random error. The effect M is assumed to be iid $N\left(0, \sigma_M^2\right)$. The error ε is assumed to be iid $N\left(0, \sigma^2\right)$ and independent of M. The effects of interest for determining differential expression across treatments would be the condition by probe interaction term PC.

1.7.2 Advantages of the Mixed Effects Model

Mixed effects models introduce added complexity to the analysis of microarray data, but also carry several potential advantages. One advantage is the greater generalizability of results (Brown and Prescott, 2006, p. 23). The fixed effects model formulates variables such as array as a separate factor. This assumes that the factors in the experiment represent all possible factors, and that additional replications of the experiment would not lead to different factors. Such an assumption would be reasonable for variables such as treatment group. However, for variable such as array, the assumption may not hold; replications of the experiment would utilize different arrays, which may not have the same characteristics as the previously used arrays. In contrast, the mixed effects model formulates variables such as array as a random sample from a larger population, with inferences applied to the population rather than the sample. Thus, if the variables follow the postulated distribution, the mixed effects model makes the inferences generalizable to a much larger group. The mixed model approach may incur some cost compared to the fixed effects model in that some effects that are significant in the latter may not be in the former, but the improved generalizability usually justifies this cost (Brown and Prescott, 2006, p. 18).

Another advantage of the mixed effects model is the reduction in the number of parameters, which improves the estimation and testing process. In the fixed effects model, specification of a variable such as array as a separate factor requires one parameter for each distinct level of the variable. In contrast, the same variable can be specified with two parameters – the mean and variance – in the mixed effects model if a normal distribution is postulated. This reduction in the number of parameters results in increased degrees of freedom for hypothesis testing, leading to more accurate estimates of effects (Brown and Prescott, 2006, chap. 2).

These two advantages are not specific to microarray data, but apply to mixed models in general. Wolfinger *et al.* (2001) proposed additional advantages of mixed effects models for microarrays. These include the extreme flexibility in experimental design and modeling; the accommodation of missing data for intensity measurements; the accommodation of arbitrary methods of background correction; and the applicability to both cDNA and oligonucleotide chips, as well as

other types of data.

Chu *et al.* (2004) illustrated the advantages of mixed effects models by comparing the mixed model of Chu *et al.* (2002) in Equation (1.14) with the fixed effects model of Li and Wong (2001) given in Equations (1.2) and (1.3). Analyses were conducted using the ionizing radiation data from Tusher *et al.* (2001) as well as simulated data. For the former, both models fit the data well, with $R^2 > 0.96$. However, for approximately 1% of the genes, the R^2 value for the mixed model was much better than that of the fixed effects model, which Chu *et al.* attributed to incorporation of a treatment by probe interaction effect and better handling of outliers in their model. The use of the mixed model also resulted in the gain of 18 degrees of freedom in testing the significance of parameters. The simulation studies suggested that the mixed effects model was associated with slightly less liberal confidence intervals and slightly better R^2 values than the fixed effects model, although both fit the data well with $R^2 > 0.92$.

1.7.3 Implementation

The mixed effects model represents a general statistical analysis technique. Although several researchers have advocated a mixed model approach to microarray data, there is currently no specific microarray analysis software dedicated to mixed effects modeling. Previous studies have used PROC MIXED in the SAS statistical system (SAS Institute, Cary, NC) for data analysis with good results. Within the R programming environment, the *limma* package (http://www.bioconductor.org) provides a variety of functions for studying microarrays with linear models, though it is primarily intended for fixed effects models and only allows one random effect at present. The R package *nlme* (http://cran.r-project.org) has a number of functions for both linear and nonlinear mixed effects models, although it has not been applied to microarray data. Facilities for mixed effects model analysis are also available in a number of other statistical packages, though their use with microarray experiments has not been specifically described.

1.8 Review of Probe-Level Analysis

Despite some variations in how analyses are conducted, all of the above methods operate at the probeset level. Prior to analysis, the probe intensities are summarized into a single measure of expression for each probeset, using one of the various expression summary algorithms. This is followed by statistical tests to determine which genes are differentially expressed between conditions. The summarization step accomplishes data reduction, a common goal in statistics. However, the efficiency of the summarization statistic is also of interest; efficiency is a measure of the variance of the statistic relative to another statistic or to the theoretical minimum variance predicted by the Cramér-Rao inequality. Lemon *et al.* (2003) provide some initial data regarding this issue. Using the Affymetrix U95 spike-in dataset, they compared the coefficient of variation calculated with probe-level data to the coefficient of variation calculated with the MAS5 and dChip expression summaries. Both of the probeset-level measures failed to reach the optimal efficiency predicted by sampling theory, indicating that the summary measures do not capture all of the information contained in the individual probes. Summarization methods are improving, and much of current research efforts center on developing more effective probeset summary measures. An alternative approach would be probe-level analysis.

For probe-level analysis, statistical testing is performed using probe-level data directly, rather than summary values. Such an approach has some potential advantages. First, by avoiding the use of expression summary measures, there is no loss of information from the summarization step (Lemon *et al.*, 2003). Second, probe-level models include the variation of probes within a probeset, which often exceeds the variation of probes across chips (Yang *et al.*, 2003). High variability between the probes within a probeset would generally lead to problems in summarizing the probe intensities with a single probeset expression value. Finally, the use of probe level data increases the information in the experiment by an order of magnitude (Barrera *et al.*, 2004), which may allow for more accurate tests of differential gene expression.

Although probe level analysis represents a viable alternative for microarray data analysis, there have also been potential disadvantages that have encouraged a focus on expression summary

measures. First were computational considerations. The early microarray datasets often required reduction to be processed by then-current computer hardware in a timely manner. However, the growth in computational power in the intervening years means that statistical analysis of probe-level data has become more feasible. A second reason would be the sizable amount of experience with expression summary algorithms. These algorithms have been improving over time, and it is possible that future versions will have greater efficiency and achieve adequate reduction of probe-level data with minimal loss of information. However, this remains largely theoretical, as the efficiency bounds for expression summary values has not been established.

1.8.1 Logit-t

There have been attempts to develop algorithms for probe level analysis, though none are commonly used like the expression summary methods. One of the earliest is the PM-only model Logit-t (Lemon *et al.*, 2003). An underlying assumption for the model is that the binding of targets to microarray probes has similar dynamics to the binding of oligonucleotides in solution, which is a first-order kinetic reaction at equilibrium. (Lemon *et al.* (2003) provide empirical evidence that such an assumption is correct.) The hybridization reaction can be depicted as

$$\text{probe} + \text{target} \rightleftarrows \text{hybrid}$$

so that the forward rate reaction $Rate_f$ and reverse rate reaction $Rate_r$ are

$$Rate_f = k_f\,[\text{probe}]\,[\text{target}]$$

$$Rate_r = k_r\,[\text{hybrid}]$$

where k_f and k_r are the respective rate constants and $[\cdot]$ denotes the concentration of a reactant. The Logit-t model assumes that the the signal intensities are proportional to the concentration of

the reactant, and so treats intensities and concentrations interchangeably. The kinetic equation for the rate of formation of the hybridization complex over time is considered the same as the change in signal intensity over time, dy/dt, yielding the equation

$$\frac{dy}{dt} = k_f\,[\text{probe}]\,[\text{target}] - k_r\,[\text{hybrid}] \tag{1.15}$$

At equilibrium, $dy/dt = 0$ and Equation (1.15) becomes

$$\frac{k_f}{k_r}\,[\text{probe}] = \frac{y}{[\text{target}]} = \frac{y}{to - y} = \frac{\dfrac{y}{to}}{1 - \dfrac{y}{to}} \tag{1.16}$$

where to is the total amount of probe available for binding. After removing the additive background signal bg, Equation (1.16) becomes

$$\frac{k_f}{k_r}\,[\text{probe}] = \frac{\dfrac{y - bg}{to - bg}}{1 - \dfrac{y - bg}{to - bg}} \tag{1.17}$$

Log-transformation of the last part of Equation (1.17) yields the equation

$$\text{logit}\,(y) = \log\left(\frac{y - bg}{to - y}\right) \tag{1.18}$$

Log-transformation of the first part of Equation (1.15) gives a second equation for $\text{logit}\,(y)$ in terms of the amount of target RNA, [target], from which Lemon *et al.* constructed a second index called Logit_Exp. However, this index did not offer better performance than other methods, and only the Logit-t was developed further. For the Logit-t model, the maximal signal intensity for the array is substituted for to and the background signal for the array is substituted for bg. Both maximal and background signal intensities are assumed to be constant for an array and are estimated by adding or subtracting 0.1% of the intensity range to the maximum and minimum probe intensities, respectively. The authors note that, after using the transformation in Equa-

tion (1.18), the probe intensities empirically follow an normal distribution. Thus, the standard Z-transformation is applied to map the logit-transformed values to a $N(0,1)$ distribution. The Student's t-test statistic is then computed for each probe across the arrays in each condition. The Logit-t for each probeset is defined as the median of the t-test statistics of all probes within the probeset. The suggested cutoff for determining differential expression with Logit-t is the t-test value corresponding to $p < 0.01$ with the degrees of freedom equal to the total number of arrays minus 2.

The original validation studies compared Logit-t to MAS5, dChip, and RMA using the Affymetrix U95 and GeneLogic U95 spike-in datasets. As each dataset contains multiple groups of chips hybridized at differing concentrations, analyses used all pairwise comparisons between concentration levels of a dataset. For each pair, the Logit-t statistic for each probeset was computed across all chips within a concentration level, and probesets were declared differentially expressed using the cutoff level of $p < 0.01$. For the other three methods, expression summary values were computed using the respective software packages. For each pair, the Student's t-test for each probeset was computed using all chips within a concentration level. A cutoff of $p < 0.01$ was considered significant for declaring a probeset to be differentially expressed. The results of all methods were then compared to the list of spike-in probesets to determine accuracy. With the Affymetrix dataset, the sensitivity of the four methods ranged from 75% to 87%; for the GeneLogic dataset the sensitivity was lower, ranging from 11% to 67%. The specificity of all methods was 99% to 100% across all analyses. However, the positive predictive value (PPV) for the Logit-t was consistently greater than the other methods, with rates of 22% to 76% for the former but only 2% to 10% for the latter. The receiver operating characteristic (ROC) curves showed similar results, with Logit-t having the highest area under the curve (AUC), followed by RMA, dChip, and MAS5. Based on these results, Lemon *et al.* concluded that probe-level analysis with the Logit-t algorithm had significant advantages over other methods by reducing the number of false positives with little increase in the number of false negatives. A software implementation of the algorithm in C was made available by the authors, and the Logit-t algorithm has been used in several peer-reviewed publications (David *et al.*, 2006; Hueber *et al.*, 2007; Nielsen *et al.*, 2006;

Balasubramanian *et al.*, 2006; Brodersen *et al.*, 2006; Mogensen *et al.*, 2006; Schwab *et al.*, 2006; Leibfried *et al.*, 2005; Wigge *et al.*, 2005; Andreasson *et al.*, 2005; Schwab *et al.*, 2005; Master *et al.*, 2005).

1.8.2 Multi-mgMOS

Liu *et al.* (2005) develop a much different formulation with their multichip modified gamma model of oligonucleotide signal (mmgMOS) model. This model utilizes information from both the PM and MM intensities and is built on the assumption that both follow a gamma distribution. Specifically, the distribution of the PM and MM intensities for probeset s, probe pair p, and array m are given by

$$PM_{msp} \sim \text{Gamma}\left(bg_{ms} + sg_{ms}, sc_{sp}\right) \tag{1.19}$$

$$MM_{msp} \sim \text{Gamma}\left(bg_{ms} + \phi sg_{ms}, sc_{sp}\right) \tag{1.20}$$

The choice of the gamma distribution is made on empirical, not theoretical, grounds. The gamma distribution is capable of modeling a variety of empirical distributions, including highly skewed ones such as that typically seen for probe intensities. In Equations (1.19) and (1.20), the constant ϕ represents the amount of specific binding present in the MM probes, which proportional to the specific binding of the PM probes. The shape parameter has two components: a nonspecific hybridization signal bg and a specific hybridization signal sg, both of which are specific to a probeset on a particular chip. The scale parameter sc is a hyperparameter that also follows a gamma distribution

$$sc_{sp} \sim \text{Gamma}\left(sh_s, ch_s\right) \tag{1.21}$$

where sh and ch are probeset-specific parameters that are shared across chips. The quantity of interest is the corrected intensities $y_{msp} = PM_{msp} - MM_{msp}$. From the properties of the gamma

distribution,

$$y_{msp} \sim \text{Gamma}\left(sg_{ms}, sc_{sp}\right)$$

The parameters in Equations (1.19) – (1.21) are estimated using an iterative maximum likelihood procedure. First, while ϕ is held fixed, bg, sg, sh, and ch are fitted. Next, with these four variables held fixed, ϕ is fitted. This process is repeated until convergence. The CDF of the specific signal is then

$$\begin{aligned} p\left(y_{msp}|sg_{ms}, sh_s, ch_s\right) &= \int p\left(y_{msp}|sg_{ms}, sc_{sp}\right) p\left(sc_{sp}|sh_s, ch_s\right) \mathrm{d}sc_{sp} \\ &= \frac{\Gamma\left(sh_s + sg_{ms}\right) ch_s^{sh_s} y_{msp}^{sg_{ms}-1}}{\Gamma\left(sg_{ms}\right)\Gamma\left(sh_s\right)\left(ch_s + y_{msp}\right)^{sh_s+sg_{ms}}} \end{aligned}$$

and the expected value and variance of the logarithm of the specific signal are

$$\log\left(y_{msp}\right) = \log\left(ch_s\right) + \Psi\left(sg_{ms}\right) - \Psi\left(sh_s\right) \tag{1.22}$$

$$\text{var}\left(\log\left(y_{msp}\right)\right) = \Psi'\left(sg_{ms}\right) + \Psi'\left(sh_s\right) \tag{1.23}$$

where $\Psi(\cdot)$ is the digamma function and $\Psi'(\cdot)$ is the first derivative of the digamma function. Equations (1.22) and (1.23) are then used to calculate percentiles and credibility intervals for y_{msp}, which determine whether probeset s is declared differentially expressed. The mmgMOS algorithm is available in the *puma* package from Bioconductor. The algorithm has been used in peer-reviewed studies, though mostly by the original authors (Liu *et al.*, 2007; Purutcuoglu and Wit, 2007; Rattray *et al.*, 2006; Liu *et al.*, 2006; Sanguinetti *et al.*, 2006).

1.8.3 Probe-level ANOVA

Barrera *et al.* (2004) provide an approach to probe-level modeling based on ANOVA. By including both probe and treatment effects, this model accounts for the variation among probes when

testing for the significance of treatments. The ANOVA model used is

$$y_{cmp} = \mu + P_p + C_c + CP_{cp} + \varepsilon_{m(cp)}$$

where y is the logarithm of the probe intensity, μ is the overall mean, P is the effect of the pth probe, C is the effect of the cth condition (treatment in the original paper), (CP) is the probe-condition interaction, and ε is the error term. Testing is performed using an F-test comparing between-treatment variation to within-treatment variation with the appropriate degrees of freedom. Barrera *et al.* also provide a nonparametric approach using the Mack-Skillings test (Mack and Skillings, 1980). This is a generalization of the Friedman test (Friedman, 1937) to include replicate data and is a distribution-free analogue of the two-way ANOVA. In this particular application, the Mack-Skillings test compares the squared deviation of the sum of the ranks for probes within a probeset to the expected sum under the null hypothesis of no differential expression. Similar to the Wilcoxon test, cutoff values for the Mack-Skillings test can be computed numerically for small sample sizes, and a χ^2 approximation can be used for larger samples.

Validation studies were carried out using the Affymetrix U95 Latin Square spike-in dataset and the FBSS dataset of Lemon *et al.* (2002). The latter dataset consists of 18 Affymetrix HuGeneFL chips in 3 conditions (serum starved, serum stimulated, and a 1:1 mixture of starved/stimulated) containing 6 replicates each. Comparisons were made using one-way tests (the parametric t-test and the nonparametric Wilcoxon test), Logit-t, and probe-level ANOVA. For the one-way tests, arrays were preprocessed using the PM-only model in dChip for background correction, normalization, and summarization. For the one-way tests and the probe-level ANOVA, p-values were adjusted for multiple testing by controlling the FDR, using the step-up procedure of Benjamini and Hochberg (1995) and a resampling-based procedure similar to the one in the SAM software (Tusher *et al.*, 2001). Comparisons of the serum-starved and the serum-stimulated chips from the FBSS dataset showed that probe-level ANOVA labeled the largest number of genes as differentially expressed. Comparisons of chips within conditions showed that no method declared genes differentially expressed, so the increased number of positive results with probe-

level ANOVA did not appear to come from increased false positives. Pairwise tests using the Affymetrix U95 spike-in dataset showed that probe-level ANOVA was capable of detecting true differential expression with a much lower fold-change than the one-way methods. Finally, Barrera *et al.* examined the tradeoff between sensitivity and specificity using ROC curves for each method on the Affymetrix data with a constant two-fold change. Probe-level ANOVA had the largest AUC, with sensitivity of 91% and specificity of 99.84%. This was followed closely by the Mack-Skillings test, while the one-way methods performed at near chance levels. Probe-level ANOVA also had the highest AUC as a function of sample size. Based on these results, the authors concluded that probe-level ANOVA offered similar performance to Logit-t, and both probe-level methods offered significant advantages over probeset-level methods. The probe-level ANOVA algorithm has been used in peer-reviewed studies (Lemieux, 2006), and a MATLAB version of the algorithm available from the authors (http://carrier.gnf.org/publications/ProbeStatistics).

1.9 Summary of Current Research

In a relatively short time frame, microarrays have advanced from a novel, unproven methodology to a commonly used technology in the analysis of differential gene expression. Parallel developments in analytical methodology have resulted in significant improvements in expression analysis algorithms. However, much of this work for Affymetrix GeneChips has focused on probeset-level expression summaries. Although such an approach has proven useful, more recent research shows that analysis of Affymetrix data at the probe level is feasible and may offer significant gains in accuracy. Furthermore, most analytical methods have utilized fixed-effects models in hypothesis testing of differential gene expression. The use of mixed effects models for Affymetrix data appears promising but has not been extensively investigated.

The central hypothesis of this proposal is that analysis of Affymetrix GeneChips using probe-level methodology will result in significant improvements in the detection of differential gene expression compared to probeset-level methods. The next two sections describe two algorithms, the S-Score and the Random Variance Model, which will be extended for probe-level testing of

multiple chips. The S-Score algorithm is a validated probe-level model for analyzing pairs of arrays, which will require extension of the model to multiple chips. The Random Variance Model is a probeset-level algorithm that borrows information across genes by modeling a distribution for the gene-specific variances; it will require extension to a probe-level model. Subsequent sections will provide the theoretical derivations underlying these extensions and details of the software implementation, followed by the results of comparisons of these probe-level methods to probeset-level methods using standardized datasets. Finally, these implications of these results for analysis of differential gene expression will be discussed, and directions for future research will be presented.

Chapter 2

The S-Score Algorithm

2.1 Development

The S-Score algorithm represents one of the earliest probe-level analysis methods (Zhang *et al.*, 2002). It is based on a novel error model for the expression of probe pair signals, in which the error for the detected signal is assumed to be proportional to the signal intensity for highly expressed genes, but approaches a background level (rather than 0) for poorly expressed genes. It has similarities to the error model of Hughes *et al.* (2000) for cDNA microarrays, which incorporates both an additive and multiplicative component. In their model, the error associated with channels 1 and 2 of gene g (called a *spot* in the original paper) is given by

$$\varepsilon_g = \sqrt{bv_{1g}^2 + bv_{2g}^2 + \gamma^2 \left(y_{1g}^2 + y_{2g}^2\right)} \tag{2.1}$$

where y is the measured signal intensity, bv is the uncertainty (variance) due to background subtraction, and γ is a fractional multiplicative error due to technical variation. The significance statistic Z for gene g on the array is then given by

$$Z_g = \frac{y_{1g} - y_{2g}}{\sqrt{bv_{1g}^2 + bv_{2g}^2 + \gamma^2 \left(y_{1g}^2 + y_{2g}^2\right)}}.$$

Hughes *et al.* reported that empirically Z was found to approximately follow a normal distribution, and bv and γ were chosen so that Z had unit variance. Also, based on empirical evidence,

they reported that bv tended to vary among arrays and derived this quantity from the background fluctuation level for each array; while γ tended to be fairly constant among arrays and was determined from control experiments where nominally the same sample was hybridized in both channels. The probability of the statistic X being as or more extreme than the one observed can then be calculated as

$$\text{p-value} = 2\left(1 - \mathbf{\Phi}\left(\left|Z_g\right|\right)\right)$$

where $\mathbf{\Phi}(\cdot)$ is the standard normal CDF.

This cDNA model was used by Zhang *et al.* to develop a similar probe-level model for oligonucleotide arrays. This model compares two conditions of one microarray each, so that $N_c = 2$ and $N_a = 1$. For their model, the error estimate for the pth probe pair of probeset s for two GeneChips 1 and 2 is given by

$$\varepsilon_{sp} = \sqrt{\gamma^2\left(y_{1sp}^2 + y_{2sp}^2\right) + bv_1^2 + bv_2^2} \tag{2.2}$$

where bv_1 and bv_2 are the background noise estimates associated with GeneChips 1 and 2, respectively; y_{1sp} and y_{2sp} are the $PM_{1sp} - MM_{1sp}$ and $PM_{2sp} - MM_{2sp}$ probe pair differences for GeneChips 1 and 2; and γ is a predefined value assumed to be constant for all GeneChips which represents the proportionality of error attributed to highly expressed genes. Therefore, γ may be thought of as the additional proportion of error attributed to y_{1sp} and y_{2sp}, which results in a larger quantity for highly abundant genes when y_{1sp} and y_{2sp} are much greater than bv_1 and bv_2.

The values of bv_1, bv_2, and γ are given by

$$\begin{aligned} bv_1 &= SDT_1 \\ &= 4 \times RawQ_1 \\ &= 4 \times \frac{1}{BG_1}\left(\sum_{i=1}^{BG_1} \frac{stdev_i}{\sqrt{pixel_i}}\right) \times SF_1 \\ bv_2 &= SDT_2 \\ &= 4 \times RawQ_2 \\ &= 4 \times \frac{1}{BG_2}\left(\sum_{i=1}^{BG_2} \frac{stdev_i}{\sqrt{pixel_i}}\right) \times SF_2 \\ \gamma &= 0.1 \end{aligned} \tag{2.3}$$

where SDT_1 and SDT_2 are the Statistical (or Standard) Difference Threshold (SDT) values of GeneChip 1 and 2, respectively. *RawQ* is an estimate of the background noise, where *BG* is the number of probes used in the background estimate; $stdev_i$ and $pixel_i$ are the standard deviation and number of pixels for the i^{th} probe among probes used in the background estimate; and the Scale Factor (SF) is used to scale each of the intensities on the chip to a specified target background value (Affymetrix, 2002b). The values of *RawQ* and *SF* are available from the Affymetrix GeneChip Operating Software. The value of γ was chosen as indicated in Equation (2.3) so that the scale of the S-Scores does not depend on the expression levels of a gene. This is consistent with the work of Hughes *et al.* for cDNA arrays.

These probe pair level error estimates are then used in the calculation of a new statistic measuring relative change in gene expression, which Zhang *et al.* called the significance score or S-Score for brevity. A relative change in probe pair intensities is calculated that converts the probe pair signal differences into multiple measurements with equalized errors. These relative changes are then summed to form the S-Score, which is a single measure of the significance of

change for the gene in question. For probe set s, the S-Score is calculated as

$$S_s = \sum_{i=1}^{N_p} \frac{y_{2sp} - y_{1sp}}{\alpha \varepsilon_{sp} \sqrt{N_p}} \tag{2.4}$$

where y_{1sp}, y_{2sp}, and ε_{sp} are as in Equation (2.2); N_p is the number of probe pairs within the sth probe set; and α is a normalization factor that corrects for the effect of correlation among probe pair signals. The value of α was chosen for an individual chip so that the variance of S-Score values on an array is 1 when outliers are excluded. For non-differentially expressed genes, Zhang *et al.* state that the S-Score follows a standard normal distribution, though their derivation is empiric rather than theoretical. P-values are readily calculated using the S-Score values, which aid in determining the significant differences in gene expression. The S-Score method eliminates the need for estimation of probe set expression summaries, simplifying the analytical process. S-Scores, by virtue of their direct comparison of individual probe-pair data, provide comparison of the expression change between two chips. This allows at least inferential statistics on experiments with limited numbers of microarrays.

The ε_{sp} in Equation (2.2) does not represent a rigorous statistical error estimate, but an intuitive proxy for this quantity. The variance of $y_{2sp} - y_{1sp}$, the difference in signal intensities between GeneChips 1 and 2, would be

$$\mathrm{var}\left(y_{1sp} - y_{2sp}\right) = \mathrm{var}\left(y_{1sp}\right) + \mathrm{var}\left(y_{2sp}\right) + bv_1^2 + bv_2^2$$

assuming that GeneChips 1 and 2 are independent, and that the standard deviation of the background for GeneChips 1 and 2 is bv_1 and bv_2 respectively. Unfortunately, the variance of y_{1sp} and y_{2sp} cannot be directly estimated as there is only one observation for the probe on each chip. Equation (2.2) utilizes

$$y_{1sp}^2 = \frac{\left(\left(PM_{1sp} - MM_{1sp}\right) - 0\right)^2}{1} \tag{2.5}$$

as a proxy variance estimate for y_{1sp} (and similarly for y_{2sp}), weighted by the factor γ. This proxy is not ideal as it would represent a biased estimate of the variance.

2.2 Implementation

The S-Score algorithm was originally coded in C++ with a compiled executable available from the authors (Kennedy *et al.*, 2006b). The algorithm was later ported to Borland Delphi, a variant of the Pascal language (Kerns *et al.*, 2003). A stand-alone GUI version for the Microsoft Windows operating system is available from http://www.brainchip.vcu.edu. This software directly reads the *.CEL files for the set of experiments and produces a tab-delimited file containing the S-Score values for each probeset, which may be read and further analyzed using standard statistical or spreadsheet programs.

More recently, Kennedy *et al.* (2006b) have implemented the S-Score algorithm in R, an open source programming environment (R Development Core Team, 2007). Integration with R allows the user to customize the analysis in ways that were not previously available, such as the use of preprocessing and visualization functions. Being open source, this implementation may be further modified to meet specific needs of individual users. The *sscore* package available through the Bioconductor (Gentleman *et al.*, 2004) project. The package has two main functions, *SScore* and *SScoreBatch*, which calculate the S-Score statistics. The former performs a two-chip comparison, while the latter performs multiple pairwise comparison of chips.

2.3 Performance Assessment

Zhang *et al.* (2002) conducted initial studies assessing the performance of the S-Score at the time of its development. In their first study, the S-Score was used to analyze expression profiles between different brain regions (prefrontal cortex versus ventral tegmentum) and similar brain regions (ventral tegmentum versus ventral tegmentum) across experimental animals. In the former, significant differences would be expected, reflecting the varying genetic expression in dissimilar

brain regions; and these differences in expression would be consistent across subjects. In the latter, any differences would be expected to represent random variation or experimental error, so these differences would be small and not consistent across subjects. This was indeed the case, with a correlation coefficient between S-Scores of $R = 0.75$ in the former case but only $R = 0.17$ in the latter case. Thus, S-Score values were clearly reproducible when comparing two distinctive brain regions, prefrontal cortex and ventral tegmentum, that would be expected to show gene expression differences.

In their second study, comparisons were also made between the S-Score and the Affymetrix MAS4 software (Affymetrix, 1999). Comparisons were between dissimilar brain regions, prefrontal cortex versus ventral tegmentum, across two experimental animals. For the purposes of their analysis, Zhang *et al.* slightly modified the MAS4 difference call, naming the revised version the DiffCall method. This method combined the mildly increased and the mildly decreased group with the increased and decreased group, respectively, due to the small number of genes in the former two categories. It also assigned the category of no change (NC) if either gene in the comparison was declared "absent" with the MAS4 absolute call, as the MAS4 difference call can be unreliable in such situations. The S-Score was converted from a continuous to a categorical measure, which Zhang *et al.* named the SCall, by setting a threshold δ for the S-Scores. The SCall label for a gene would be increasing (I) if the S-Score for that gene is greater than δ, decreasing (D) if the S-Score is less than $-\delta$, and no change (NC) otherwise. For both the SCall and DiffCall methods, genes that were consistently classified across the two samples (i.e. increased in both samples or decreased in both samples) were labeled congruent changes, while those not so classified were labeled non-congruent changes. They also defined the consistency ratio as the number of congruent changes divided by the number of non-congruent changes, after excluding genes classified as no change. This ratio was calculated for both methods to quantify the performance in classifying gene expression changes. The consistency ratio of the S-Score across a variety of thresholds δ uniformly exceeded that of the DiffCall. Thus, the S-Score offered much greater consistency than the MAS4 algorithm, but without loss of sensitivity.

In their third study, the S-Score was compared to the logarithm of the fold-change ratio,

$\ln(Fc) = \ln(AvgDiff_2/AvgDiff_1)$ for gene expression under conditions 1 and 2. Comparisons were made between dissimilar brain regions, prefrontal cortex versus ventral tegmentum, across four experimental animals. In calculating the $\ln(Fc)$, Zhang *et al.* excluded from the calculations all genes that were declared absent across all eight of the samples. Next, all AvgDiff values that were below the background noise level were set equal to the background noise. Finally, they excluded all genes with $|\ln(Fc)| < \ln(2)$, or less than two-fold change, across all prefrontal cortex versus ventral tegmentum comparisons. The remaining 867 genes were then subjected to average linkage cluster analysis. The S-Score was found to produce "tighter" clusters than the $\ln(Fc)$, with the average correlation coefficient across clusters of $R = 0.91$ for the former and $R = 0.70$ for the latter ($p < 10^{-10}$, Fisher's Z-transformation). Similarly, the correlation coefficient between experimental animals for the S-Score was much higher ($R = 0.91$) than that of $\ln(Fc)$ ($R = 0.71$), again without loss of sensitivity. Finally, the S-Score and $\ln(Fc)$ were compared for four brain regions, prefrontal cortex, ventral tegmentum, hippocampus, and nucleus accumbens, across four experimental animals using cluster analysis. For the $\ln(Fc)$, genes with a MAS4 absolute call of "absent" across all 16 samples and genes having a variance of $\ln(Fc)$ below an unspecified arbitrary threshold were excluded from the analysis. The S-Scores were generated by comparing each of the 16 samples to an "average" CEL file created by averaging the intensities on the probe level across the 16 samples. Both the S-Score and the $\ln(Fc)$ produced initial clusters that grouped samples by brain region. However, the S-Score produced tighter clusters, more accurately reflecting the homogeneity within a brain region and heterogeneity between brain regions. The average correlation between samples within a brain region was 0.80 for the S-Score and 0.52 for $\ln(Fc)$. The S-Score also had higher correlation between genes within a cluster; the number of genes with a highly correlated nearest-neighbor, defined as $R > 0.90$, was 302 for the S-Score but 274 for $\ln(Fc)$. A total of 231 genes with differential expression between prefrontal cortex and ventral tegmentum were identified using a cutoff of the absolute value of the S-Score exceeding 2.5. Of the 29 showing the most distinctive changes, 20 of these had previous studies in the literature (mostly using *in situ* hybridization (ISH)) to corroborate the finding of differential expression. Thus, the S-Score also offered greater specificity than the $\ln(Fc)$, but without loss of

sensitivity.

Kerns *et al.* (2003) replicated and extended some of these findings in additional validation studies. They compared expression profiles between different brain regions (prefrontal cortex versus nucleus accumbens) and similar brain regions (nucleus accumbens versus nucleus accumbens) across experimental animals, similar to the first validation study of Zhang *et al.* Again, there were significant reproducible differences in the former case, with correlation coefficient of $R^2 = 0.65$; but little reproducible differences in the latter case, with correlation coefficient of $R^2 = 0.00002$. They also compared the S-Score to the $\log_2$-fold change produced by the Affymetrix MAS5 algorithm. Although many genes showed similar changes with the S-Score and MAS5, a number of genes with large fold change values had S-Scores near zero. Subsequent analysis showed that these genes were largely declared "absent", indicating the greater specificity of the S-Score values compared to MAS5.

Kennedy *et al.* (2006a) provide further validation of the S-Score by comparing it to RMA, dChip, and MAS5. Data for this study were drawn from two spike-in datasets publicly available from GeneLogic, Inc. using the human U95 GeneChip (http://www.genelogic.com/newsroom/studies/). For both datasets, a common complex complementary ribonucleic acid (cRNA) derived from an acute myeloid leukemia (AML) tumor cell line was used as background. Clones from different regions of 4 bacterial genes (BioB, BioC, BioD, and Dap) and of 1 phagemid gene (Cre) were then spiked into the samples at known concentrations. For the first dataset, called the Dilution dataset, 10 different clones were spiked into the hybridization cocktail at the same concentration on each array, with the 11th clone (CreX-3) used as a control at 0pM across arrays (Table 2.1). There were 1 to 3 replicates for each concentration level. For the second dataset, called the AML Latin Square dataset, 11 clones were spiked into the hybridization cocktail using different concentrations arranged in a Latin Square design (Table 2.2). There were 2 to 3 replicates at each concentration level. The use of spike-in datasets permits assessment when the true conditions of differential expression are known, a definite advantage in comparing the performance of algorithms.

For the Dilution dataset, comparisons of sensitivity and specificity were obtained by plotting

Table 2.1: Concentration Data for the GeneLogic Dilution Dataset

	Transcript										
Experiment Number	BioB-5_at	BioB-M_at	BioB-3_at	BioC-5_at	BioC-3_at	BioDn-3_at	DapX-5_at	DapX-M_at	DapX-3_at	CreX-5_at	CreX-3_at
1	0	0	0	0	0	0	0	0	0	0	0
2	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0
3	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0.75	0
4	1	1	1	1	1	1	1	1	1	1	0
5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	0
6	2	2	2	2	2	2	2	2	2	2	0
7	3	3	3	3	3	3	3	3	3	3	0
8	5	5	5	5	5	5	5	5	5	5	0
9	12.5	12.5	12.5	12.5	12.5	12.5	12.5	12.5	12.5	12.5	0
10	25	25	25	25	25	25	25	25	25	25	0
11	50	50	50	50	50	50	50	50	50	50	0
12	75	75	75	75	75	75	75	75	75	75	0
13	100	100	100	100	100	100	100	100	100	100	0
14	150	150	150	150	150	150	150	150	150	150	0

Experiment Number	Transcript									
	BioB-5_at	BioB-M_at	BioB-3_at	BioC-5_at	BioC-3_at	BioDn-3_at	DapX-5_at	DapX-M_at	DapX-3_at	CreX-5_at	CreX-3_at
1	0.5	35.7	25	75	100	50	1.5	1	3	2	5
2	1	50	35.7	100	3	75	2	1.5	5	25	12.5
3	1.5	75	50	3	5	100	25	2	12.5	35.7	0.5
4	2	100	75	5	12.5	3	35.7	25	0.5	50	1
5	3	1.5	1	25	35.7	2	12.5	5	50	0.5	75
6	5	2	1.5	35.7	50	25	0.5	12.5	75	1	100
7	12.5	25	2	50	75	35.7	1	0.5	100	1.5	3
8	37.5	5	3	0.5	1	12.5	75	50	1.5	100	2
9	50	12.5	5	1	1.5	0.5	100	75	2	3	25
10	75	0.5	12.5	1.5	2	1	3	100	25	5	35.7
11	100	1	0.5	2	25	1.5	5	3	35.7	12.5	50

Table 2.2: Concentration Data for the GeneLogic AML Latin Square Dataset

the S-Score values against the other three algorithms. Ideally, algorithms would show a clear separation between the values for the spike-in genes and remaining genes, indicating that the algorithm can distinguish between the two conditions. Such distinctions are expected to be less clear at lower RNA concentrations due to the diminished signal from the spike-in genes. The S-Score clearly separated the spike-in clones from the other probe sets at concentrations of 3pM and greater, with some loss of accuracy at lower concentrations. The RMA expression summary values also showed clear separation, although this did not occur until concentrations of 12.5pM and greater (Figure 2.1). The MBEI values produced by dChip did not show total separation at any concentration, though performance was notably better with concentrations of 5pM and greater (Figure 2.2). MAS5 p-values did not provide total separation at any concentration (Figure 2.3).

For the AML Latin Square dataset, the observed ranks of the spike-in clones were compared to their true underlying rank, based on the known fold-change, for each algorithm. This requires designation of one chip as a baseline to which all other chips are compared in the fold-change calculation. Chip 92561 was selected as the reference array, though similar results were obtained when other chips were used as a reference. Ideally, the observed rank should equal the true rank, and an algorithm would rank all 11 spike-in clones in the top 11 genes declared differentially expressed. The proportion of spike-in clones ranked in the top 11 for each method was calculated, and the proportions for the S-Score were compared to those of the other three algorithms using the Cochran-Mantel-Hanzel test (Table 2.3). The MAS5 algorithm had the highest proportion of clones ranked in the top 11, but also exhibited excessively high sensitivity, with difficulty separating the clones from other genes despite obvious differences in fold-change. The differences between the two algorithms was not statistically significant, $\chi^2(1) = .40, p > 0.52$. The observed ranks for the S-Score were generally much closer to the true underlying ranks than the ranks for RMA and dChip. The proportion of clones ranked in the top 11 was also significantly higher ($\chi^2(1) = 17.88, p < 0.001$ compared to RMA, $\chi^2(1) = 21.33, p < 0.001$ compared to dChip). After three chips of questionable quality were excluded, the performance of the S-Score was similar to that of RMA and MAS5 ($p > 0.51$ for the former and $p > 0.26$ for the latter). Taken together, these two spike-in studies showed that the S-Score offers excellent sensitivity and

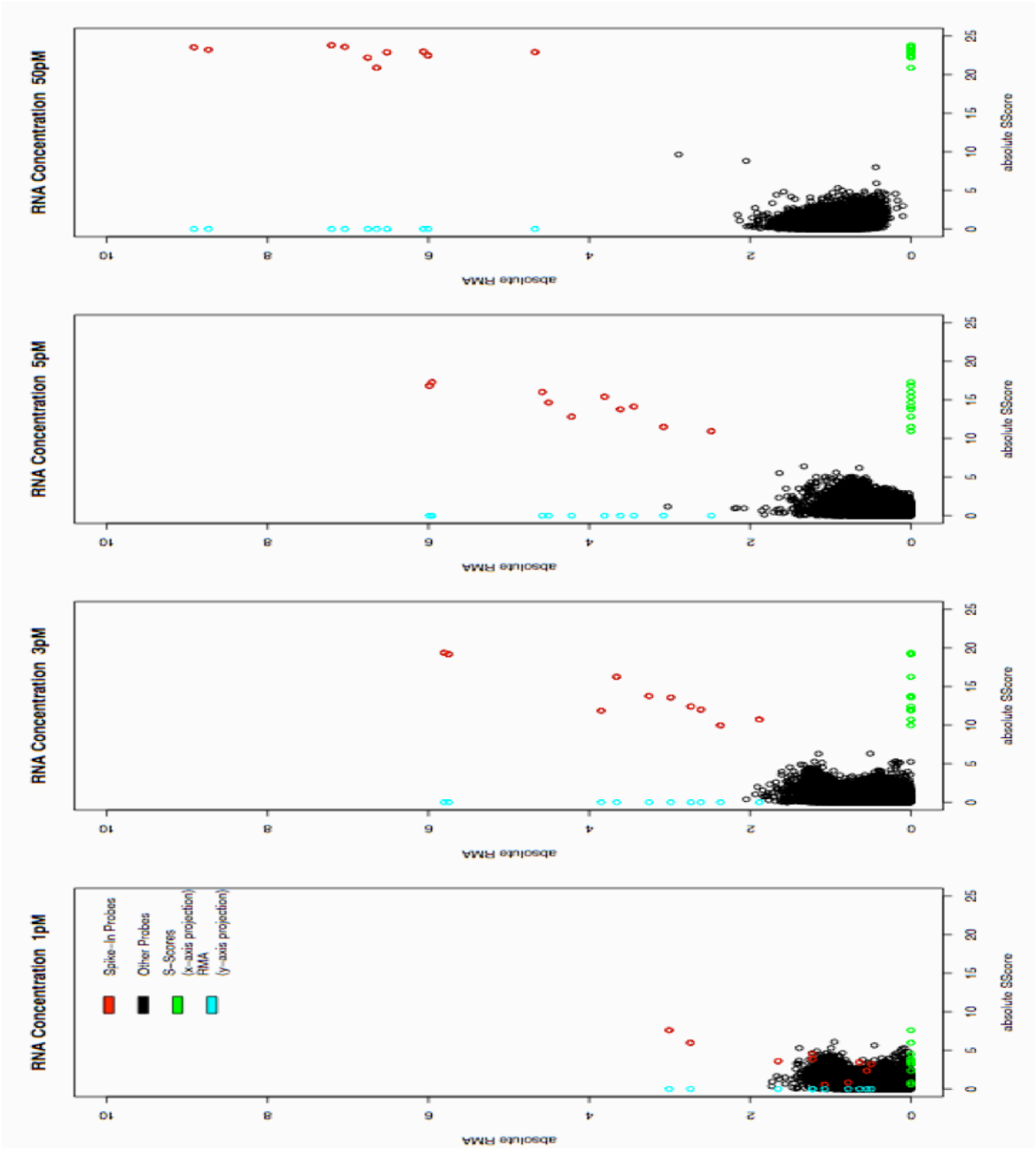

Figure 2.1: Comparison of S-Score and RMA. Plot of absolute value of S-Score vs absolute value of difference in RMA expression summaries, comparing the specified concentration to the baseline chip. X- and Y-axis projections are added to show separation of spike-in probes more clearly. From Kennedy *et al.* (2006a), used with permission.

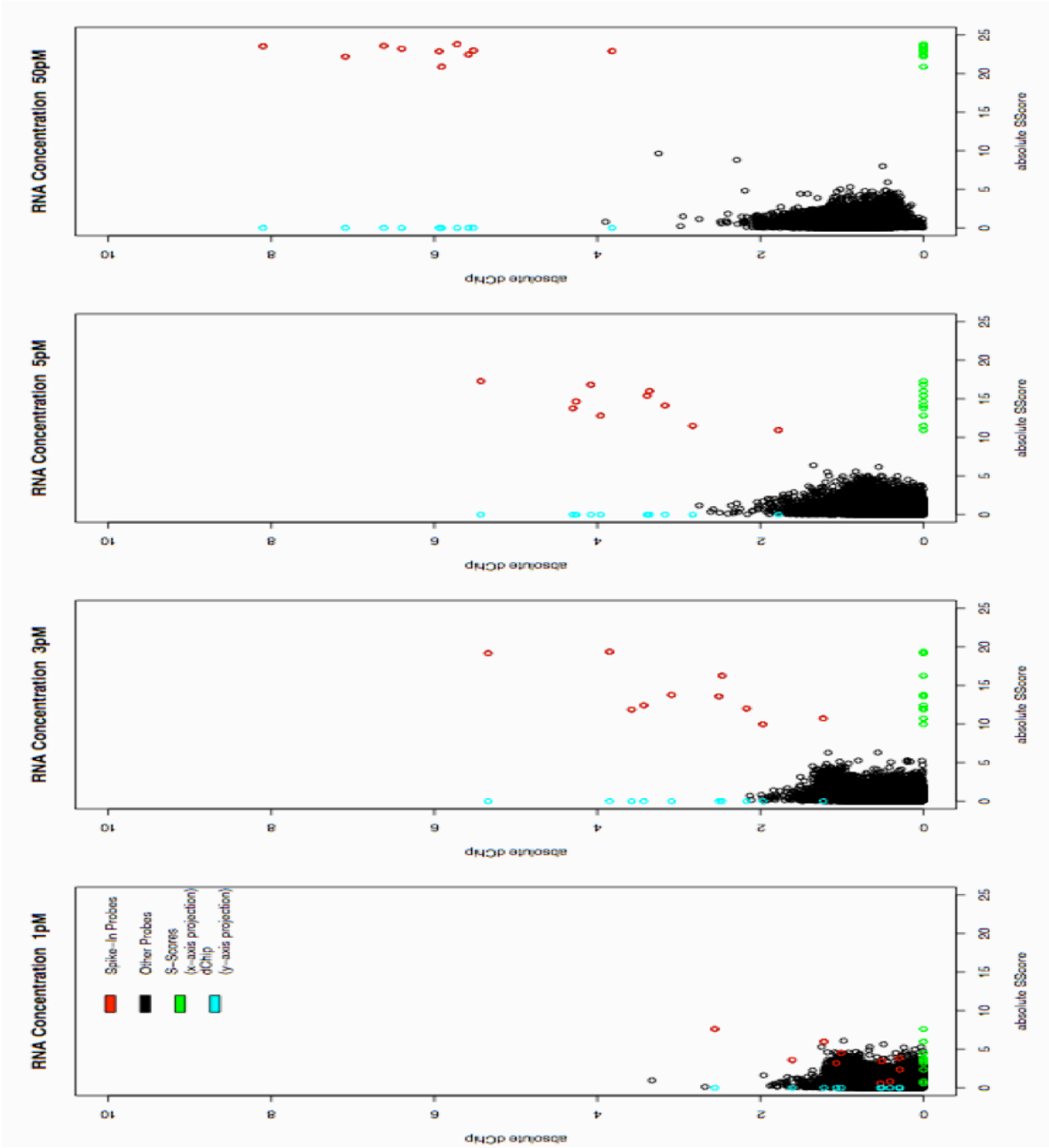

Figure 2.2: Comparison of S-Score and dChip. Plot of absolute value of S-Score vs absolute value of difference in base 2 logarithm of dChip model-based expression index, comparing the specified concentration to the baseline chip. X- and Y-axis projections are added to show separation of spike-in probes more clearly. From Kennedy *et al.* (2006a), used with permission.

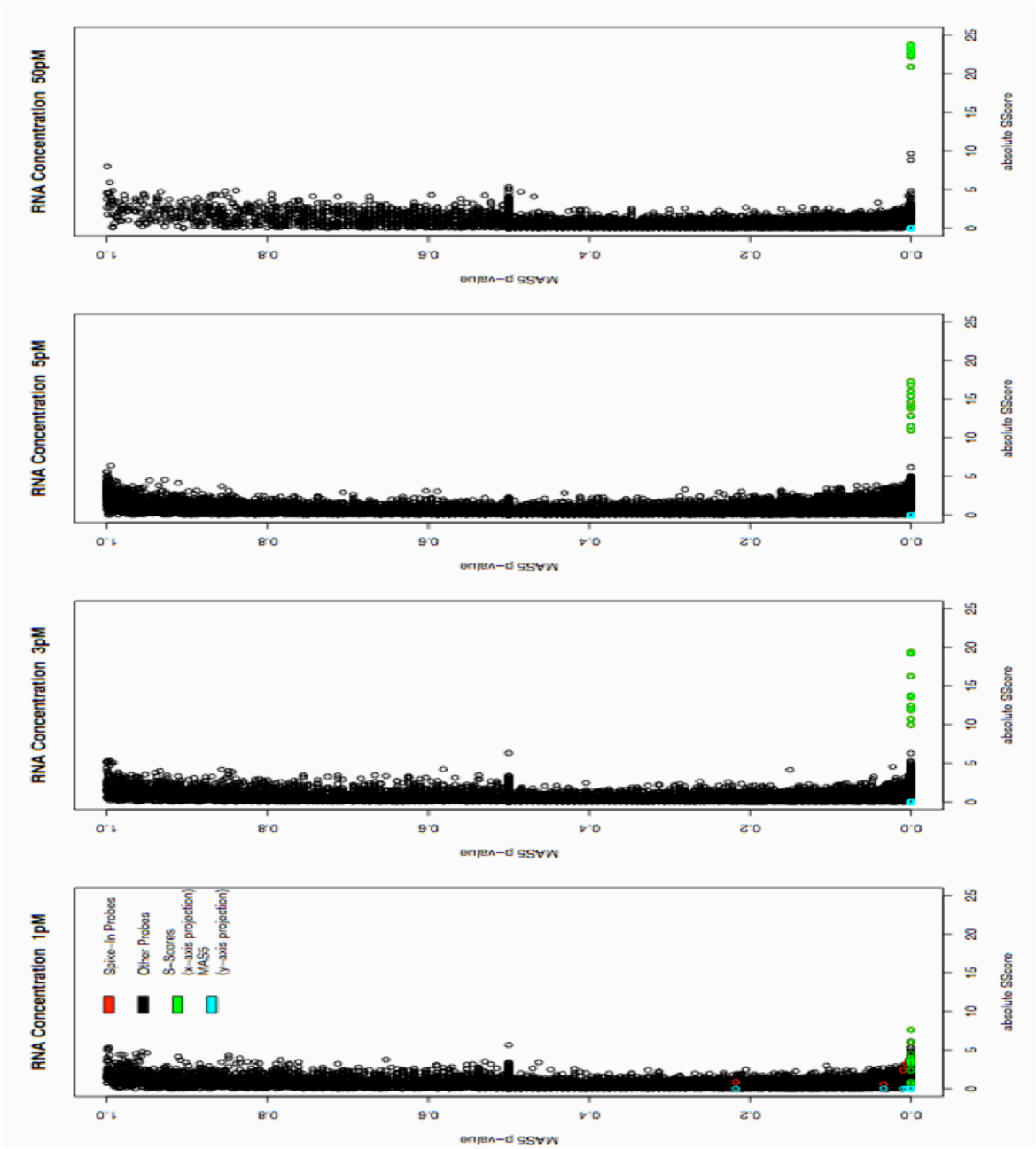

Figure 2.3: Comparison of S-Score and MAS5. Plot of absolute value of S-Score vs MAS5 p-values, comparing the specified concentration to the baseline chip. MAS5 p-values were transformed so that significantly up- and down-regulated genes will have p-values approaching 0. X- and Y-axis projections are added to show separation of spike-in probes more clearly. From Kennedy *et al.* (2006a), used with permission.

GeneChip Array	S-Score	RMA	dChip	MAS5
92562	4 (0.36)	4 (0.36)	4 (0.36)	4 (0.36)
92563	8 (0.72)	4 (0.36)	4 (0.36)	7 (0.64)
92564	8 (0.72)	0 (0.00)	2 (0.18)	8 (0.72)
92558	10 (0.90)	10 (0.90)	8 (0.72)	11 (1.00)
92559	11 (1.00)	9 (0.81)	9 (0.81)	11 (1.00)
92560	9 (0.81)	7 (0.63)	7 (0.63)	9 (0.81)
92554	10 (0.90)	8 (0.72)	6 (0.54)	11 (1.00)
92555	11 (1.00)	9 (0.81)	9 (0.81)	11 (1.00)
92556	10 (0.90)	9 (0.81)	7 (0.63)	11 (1.00)
92557	10 (0.90)	5 (0.45)	5 (0.45)	11 (1.00)

Table 2.3: Number and proportion of spike-in clones detected using chip 92561 as baseline. Comparison of S-Score vs. RMA, $p < 0.001$; vs. dChip, $p < 0.001$; vs. MAS5, $p = 0.40$.

specificity compared to RMA, dChip, and MAS5.

2.4 Limitations

While the S-Score algorithm successfully implements probe-level analysis for Affymetrix data, limitations are present. The original implementation only allows comparisons between pairs of microarray chips. Although many early studies only had single chips for analyzing each condition, replicate data are necessary for discerning true gene expression changes from artifacts. Current guidelines suggest three or more chips per condition to achieve satisfactory results. For the S-Score algorithm to be a useful method in the analysis of microarray data, it must be capable of analyzing experiments following these recommendations and incorporating replicate chips. At the same time, most experiments with replicates still have relatively small sample sizes (Jain *et al.*, 2003), so the S-Score algorithm must still achieve substantial accuracy with only a small number of chips.

A second potential limitation concerns the error model used in the calculation of the S-Score. Although the accuracy of this model appears to be good, two of the key parameters, γ and bv, are *ad hoc* in nature rather than mathematically rigorous. It is hoped that a more rigorously derived analogue for these quantities would give even greater accuracy in the detection of differential

gene expression. Furthermore, if the newly defined error model should prove inferior the the one currently used in the S-Score, this would prompt further investigation to discover the relevant biological characteristics of microarrays that must be incorporated into existing mathematical models.

Chapter 3

The Random Variance Model Algorithm

3.1 Development

The Random Variance Model algorithm (Wright and Simon, 2003) was developed as a method that may yield more accurate estimates of gene expression variances, and hence test statistics, for microarray datasets differences with small sample sizes. The RVM is based on the assumption that the variances of the intensity measurements vary from probeset to probeset, but represent random selections from a common distribution. Such an approach allows sharing of information across probesets and is expected to produce more accurate estimates for use in hypothesis testing.

The RVM algorithm utilizes a GLM framework for testing. Let y_{ms} represent the appropriate intensity measurement for microarray m and probeset s. The preprocessing steps for these intensities are not specified, although they may be normalized or log-transformed if desired. The GLM for the intensities is

$$\underset{1\times1}{y_{ms}} = \underset{1\times k}{\boldsymbol{x}'_m}\,\underset{k\times1}{\boldsymbol{\beta}_s} + \underset{1\times1}{\varepsilon_{ms}}$$

where each $\boldsymbol{x}$ is a vector of design variables for the array m, each $\boldsymbol{\beta}_s$ is a vector of k unknown coefficients for the probeset s, and ε_{ms} is a random error term. The design vector $\boldsymbol{x}$ and the coefficient vector $\boldsymbol{\beta}$ would encompass the probeset effect, array effect, condition effect, and other

such terms in the GLMs previously discussed. In vector terms, the GLM is

$$\underset{n\times 1}{\boldsymbol{y}} = \underset{n\times k}{\boldsymbol{X}'}\ \underset{k\times 1}{\boldsymbol{\beta}} + \underset{n\times 1}{\boldsymbol{\varepsilon}}\,. \tag{3.1}$$

In simple linear regression, it is assumed $\varepsilon \sim N\left(0, \sigma^2\right)$, where σ^2 is a constant term for all levels of x. The underlying assumption for RVM is

$$\varepsilon_{ms} \sim N\left(0, \sigma_s^2\right)$$

where here, the σ_s^2 are realizations of the random variable σ^2 having an inverse gamma distribution, or more formally

$$\sigma^{-2} \sim \text{Gamma}\left(x; a, b\right) \equiv \frac{1}{\Gamma\left(a\right) b^a} x^{a-1} \exp\left(-x/b\right) \tag{3.2}$$

where the unknown hyperparameters a and b are estimated from the data. Although the formulation of RVM is frequentist, it should be noted that the inverse gamma distribution is also the usual choice as a prior for variances in Bayesian analysis.

The assumptions regarding the distribution of σ^2 leads naturally to a modification of the sums of squares for traditional hypothesis tests, as shown in the following theorems. Furthermore, it may be shown that the sample data can be used to obtain estimates of the hyperparameters a and b. Since the original proofs for RVM are given in supplemental material, rather than in the published manuscript, and contain some typographical errors (G. Wright, personal communication), a detailed derivation is presented below.

Using the model in Equation (3.1), the hypotheses to be tested are $H_0 : \boldsymbol{\beta} \in \omega$ versus $H_1 : \boldsymbol{\beta} \in \mathbb{R}^k$, where ω is a linear subspace of $\mathbb{R}^k$. The null hypothesis imposes $r \equiv k - \dim\left(\omega\right)$ linear constraints on the model. In standard general linear models, where there are no distributional

assumptions on σ, the maximum likelihood estimate (MLE) for $\boldsymbol{\beta}$ under H_1 is

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\,\boldsymbol{X}'\boldsymbol{y}.$$

Under H_0, the MLE is

$$\widehat{\widehat{\boldsymbol{\beta}}} = (\boldsymbol{X}'_{\omega}\boldsymbol{X}_{\omega})^{-1}\,\boldsymbol{X}'_{\omega}\boldsymbol{y}$$

where $\boldsymbol{X}_{\omega}$ is the design matrix $\boldsymbol{X}$ projected into the subspace ω. The modified test under RVM is then given by the following theorem by Wright and Simon. As in the original manuscript, the subscript s will be suppressed throughout the proof for simplicity.

Theorem 3.1. *Under RVM, the likelihood ratio test statistic $\widetilde{F}$ for testing $H_0 : \boldsymbol{\beta} \in \omega$ versus $H_1 : \boldsymbol{\beta} \notin \omega$ is of the form,*

$$\widetilde{F} = \frac{n-k+2a}{r}\left(\frac{\widehat{\widehat{SS}} - \widehat{SS}}{\widetilde{SS}}\right) \tag{3.3}$$

where $\widehat{\widehat{SS}}$, $\widehat{SS}$, and $\widetilde{SS}$ are the sums of squares under H_0, H_1, and RVM, respectively, defined as

$$\widehat{SS} = \left(\boldsymbol{y} - \boldsymbol{X}'\widehat{\boldsymbol{\beta}}\right)'\left(\boldsymbol{y} - \boldsymbol{X}'\widehat{\boldsymbol{\beta}}\right),$$

$$\widehat{\widehat{SS}} = \left(\boldsymbol{y} - \boldsymbol{X}'\widehat{\widehat{\boldsymbol{\beta}}}\right)'\left(\boldsymbol{y} - \boldsymbol{X}'\widehat{\widehat{\boldsymbol{\beta}}}\right), \tag{3.4}$$

and

$$\widetilde{SS} = \widehat{SS} + 2b^{-1}.$$

Proof. The density function of $\boldsymbol{y}$ under the linear model is

$$f(\boldsymbol{y} \mid \boldsymbol{\beta}, \sigma) = \left(\frac{1}{2\pi\sigma^2}\right)^{n/2} \exp\left(-\frac{1}{2\sigma^2}(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})'(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})\right)$$

Letting $z = \sigma^{-2}$ and including the distributional assumptions on σ^{-2} as presented in Equation (3.2), this becomes

$$\begin{aligned} f(y \mid \beta, z) &= \left(\frac{z}{2\pi}\right)^{n/2} \exp\left(-\frac{z}{2}(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})'(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})\right)\Gamma(z; a, b) \\ &= \left(\frac{z}{2\pi}\right)^{n/2} \exp\left(-\frac{z}{2}(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})'(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})\right)\frac{z^{a-1}\exp(-z/b)}{\Gamma(a)\, b^a}. \end{aligned}$$

Note that the supplemental material accompanying the original manuscript contains a typographical error, with the exponent of the first term given as 2 rather than $n/2$; this affects some intermediate results but it does not affect the final result of the theorem. This simplifies to

$$\begin{aligned} f(\boldsymbol{y} \mid \boldsymbol{\beta}, z) &= \left(\frac{1}{2\pi}\right)^{n/2} \frac{1}{\Gamma(a)\, b^a} z^{n/2+a-1} \exp\left(-z\left(\frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})'(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta}) + b^{-1}\right)\right) \\ &= c_1 z^{(n+2a-2)/2} \exp\left(-z\left(\frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})'(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta}) + b^{-1}\right)\right) \end{aligned}$$

where the term

$$c_1 = \left(\frac{1}{2\pi}\right)^{n/2} \frac{1}{\Gamma(a)\, b^a}$$

is a scaling constant.

The joint likelihood for $\boldsymbol{\beta}$ and z is

$$L(\boldsymbol{\beta}, z \mid \boldsymbol{y}) = c_1 z^{(n+2a-2)/2} \exp\left(-z\left(\frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})'(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta}) + b^{-1}\right)\right) \tag{3.5}$$

with the log likelihood

$$\log L(\boldsymbol{\beta}, z \mid \boldsymbol{y}) = c_1 + \frac{n + 2a - 2}{2} \log z - \frac{z}{2} (\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})' (\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta}) - zb^{-1}. \tag{3.6}$$

Differentiation yields

$$\frac{\partial}{\partial \boldsymbol{\beta}} \log L(\boldsymbol{\beta}, z \mid \boldsymbol{y}) = -\frac{z}{2} (0 - 2\boldsymbol{X}'\boldsymbol{y} + 2\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta}) \tag{3.7}$$

and

$$\frac{\partial}{\partial z} \log L(\boldsymbol{\beta}, z \mid \boldsymbol{y}) = \frac{n + 2a - 2}{2z} - \frac{1}{2} (\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})' (\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta}) + b^{-1}. \tag{3.8}$$

To obtain the maximum likelihood estimates $\widehat{\widehat{\boldsymbol{\beta}}}$ and $\widehat{\widehat{z}}$ under H_0, setting Equation (3.7) to $\mathbf{0}$ implies

$$-2\boldsymbol{X}'_{\omega}\boldsymbol{y} + 2\boldsymbol{X}'_{\omega}\boldsymbol{X}_{\omega}\widehat{\widehat{\boldsymbol{\beta}}} = \mathbf{0}$$

$$\widehat{\widehat{\boldsymbol{\beta}}} = (\boldsymbol{X}'_{\omega}\boldsymbol{X}_{\omega})^{-1} \boldsymbol{X}'_{\omega}\boldsymbol{y}.$$

Under H_0, setting Equation (3.8) to 0 implies

$$\frac{n + 2a - 2}{2\widehat{\widehat{z}}} = \frac{1}{2} (\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})' (\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta}) - b^{-1}$$

$$\frac{n + 2a - 2}{2\widehat{\widehat{z}}} = \frac{(\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})' (\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta}) - 2b^{-1}}{2}.$$

Cross-multiplication yields

$$2(n + 2a - 2) = 2\widehat{\widehat{z}}\left((\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta})' (\boldsymbol{y} - \boldsymbol{X}'\boldsymbol{\beta}) - 2b^{-1}\right).$$

Substituting $\widehat{\widehat{SS}}$ from Equation (3.4) and solving for $\widehat{\widehat{z}}$ gives

$$\widehat{\widehat{z}} = \frac{n + 2a - 2}{\widehat{\widehat{SS}} + 2b^{-1}}.$$

The corresponding maximum likelihood is

$$\begin{aligned}\max_{z \in R^+, \boldsymbol{\beta} \in \omega} L(\boldsymbol{\beta}, z \mid \boldsymbol{y}) &= \left(\frac{1}{2\pi}\right)^{n/2} \frac{1}{\Gamma(a)\, b^a} \exp\left(-\frac{n + 2a - 2}{2}\right)\left(\frac{n + 2a - 2}{\widehat{\widehat{SS}} + 2b^{-1}}\right)^{(n+2a-2)/2} \\ &= c_2 \left(\frac{n + 2a - 2}{\widehat{\widehat{SS}} + 2b^{-1}}\right)^{(n+2a-2)/2}\end{aligned}$$

where

$$c_2 = \left(\frac{1}{2\pi}\right)^{n/2} \frac{1}{\Gamma(a)\, b^a} \exp\left(-\frac{n + 2a - 2}{2}\right)$$

is a constant. Similarly, under H_1, the quantity in Equation (3.6) is maximized when

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1} \boldsymbol{X}'\boldsymbol{y}$$

and

$$\widehat{z} = \frac{n + 2a - 2}{\widehat{SS} + 2b^{-1}}$$

with the corresponding maximum likelihood

$$\begin{aligned}\max_{z \in R^+, \boldsymbol{\beta} \in \mathbb{R}^k} L(\beta, z \mid y) &= \left(\frac{1}{2\pi}\right)^{n/2} \frac{1}{\Gamma(a)\, b^a} \exp\left(-\frac{n + 2a - 2}{2}\right)\left(\frac{n + 2a - 2}{\widehat{SS} + 2b^{-1}}\right)^{(n+2a-2)/2} \\ &= c_2 \left(\frac{n + 2a - 2}{\widehat{SS} + 2b^{-1}}\right)^{(n+2a-2)/2}\end{aligned}$$

where c_2 is a constant.

The likelihood ratio test is then

$$\frac{\max\limits_{z\in R^+,\boldsymbol{\beta}\in\mathbb{R}^k} L(\boldsymbol{\beta}, z \mid \boldsymbol{y})}{\max\limits_{z\in R^+,\boldsymbol{\beta}\in\omega} L(\boldsymbol{\beta}, z \mid \boldsymbol{y})} > \delta$$

which reduces to

$$\left(\frac{\widehat{\widehat{SS}} + 2b^{-1}}{\widehat{SS} + 2b^{-1}}\right)^{(n+2a-2)/2} > \delta.$$

This is equivalent to

$$\frac{\widehat{\widehat{SS}} + 2b^{-1}}{\widehat{SS} + 2b^{-1}} > \delta^{2/(n+2a-2)}. \tag{3.9}$$

Subtracting $\left(\widehat{SS} + 2b^{-1}\right) / \left(\widehat{SS} + 2b^{-1}\right)$ from both sides of Equation (3.9) yields

$$\frac{\widehat{\widehat{SS}} + 2b^{-1} - \left(\widehat{SS} + 2b^{-1}\right)}{\widehat{SS} + 2b^{-1}} > \delta^{2/(n+2a-2)} - 1,$$

which simplifies to

$$\frac{\widehat{\widehat{SS}} - \widehat{SS}}{\widetilde{SS}} > \delta^{2/(n+2a-2)} - 1.$$

Thus the test rejects if

$$\widetilde{F} = \frac{n-k+2a}{r}\left(\frac{\widehat{\widehat{SS}} - \widehat{SS}}{\widetilde{SS}}\right) > \frac{n-k+2a}{r}\left(\delta^{2/(n+2a-2)} - 1\right); \tag{3.10}$$

therefore $\widetilde{F}$ is a likelihood ratio test statistic. □

However, to use Equation (3.10) for testing under the general linear hypothesis requires addi-

tional information. Define

$$\widehat{\sigma}^2 = \frac{\widehat{SS}}{n-k}$$

and

$$\widetilde{\sigma}^2 = \frac{\widetilde{SS}}{n-k+2a}.$$

For GLM with no assumptions on σ^2, $(n-k)\,\widehat{\sigma}^2/\sigma^2$ is distributed as χ^2_{n-k} and is statistically independent of the value in the numerator of the ratio in Equation (3.10). (Note that the supplemental material contains a typographical error, stating that $(n-k)\,\widetilde{\sigma}^2/\sigma^2$ follows this distribution rather than $(n-k)\,\widehat{\sigma}^2/\sigma^2$.) It immediately follows that $\widetilde{\sigma}^2$ is statistically independent of the numerator in Equation (3.10), as $\widetilde{\sigma}^2$ depends on the data only through $\widehat{\sigma}^2$. The next theorem gives the distribution of $\widetilde{\sigma}^2$ and shows how estimates of a and b may be found using the sample data.

Theorem 3.2. *For $\widehat{\sigma}^2$ and $\widetilde{\sigma}^2$ as above,*

$$(n-k+2a)\,\frac{\widetilde{\sigma}^2}{\sigma^2} \sim \chi^2_{n-k+2a}$$

and

$$ab\left(\widetilde{\sigma}^2_s\right) \sim F_{n-k,2a}. \tag{3.11}$$

Proof. From standard ANOVA theory, for fixed σ^2

$$\left((n-k)\,\frac{\widehat{\sigma}^2}{\sigma^2}\right) \sim \chi^2_{n-k}.$$

Let $K = n-k$. Using a standard change of variables formula from $K\widehat{\sigma}^2/\sigma^2$ to $\widehat{\sigma}^2$, the distribution

of $\widehat{\sigma}^2$ is

$$\widehat{\sigma}^2 \sim \frac{K}{\sigma^2}\chi^2_K\left(\frac{K\widehat{\sigma}^2}{\sigma^2}\right).$$

The density function of $\widehat{\sigma}^2$ is

$$f\left(\widehat{\sigma}^2\right) = \frac{K}{\sigma^2\Gamma\left(K/2\right)2^{K/2}}\left(\frac{K\widehat{\sigma}^2}{\sigma^2}\right)^{K/2-1}\exp\left(\frac{-K\widehat{\sigma}^2}{2\sigma^2}\right).$$

Let $z = \sigma^{-2}$. Under RVM assumptions, $z \sim \text{Gamma}\left(a, b\right)$, so that the joint density of $\widehat{\sigma}^2$ and z is

$$\begin{aligned} f\left(\widehat{\sigma}^2, z\right) &= f\left(\widehat{\sigma}^2|z\right) f\left(z\right) \\ &= \frac{zK}{\Gamma\left(K/2\right)2^{K/2}}\left(zK\widehat{\sigma}^2\right)^{K/2-1}\exp\left(\frac{-zK\widehat{\sigma}^2}{2}\right)\left(\frac{z^{a-1}\exp\left(-z/b\right)}{\Gamma\left(a\right)b^a}\right) \end{aligned}$$

(Note that the supplemental material contains a typographical error, referring to the joint distribution of σ^2 and z rather than $\widehat{\sigma}^2$ and z.)

Removing the constant terms and grouping the z and exponential terms gives

$$f\left(\widehat{\sigma}^2, z\right) \propto \left(\widehat{\sigma}^2\right)^{K/2-1}\exp\left(-z\left(\frac{K}{2}\widehat{\sigma}^2 + b^{-1}\right)\right)\left(z^{K/2+a-1}\right).$$

Then

$$f\left(\widehat{\sigma}^2, z\right) \propto \left(\widehat{\sigma}^2\right)^{K/2-1}\left(\frac{K}{2}\widehat{\sigma}^2 + b^{-1}\right)^{1-(K/2+a)}\exp\left(-z\left(\frac{K}{2}\widehat{\sigma}^2 + b^{-1}\right)\right)\left(z\left(\frac{K}{2}\widehat{\sigma}^2 + b^{-1}\right)\right)^{K/2+a-1} \tag{3.12}$$

since

$$\begin{aligned} \left(\frac{K}{2}\widehat{\sigma}^2 + b^{-1}\right)^{1-(K/2+a)}\left(z\left(\frac{K}{2}\widehat{\sigma}^2 + b^{-1}\right)\right)^{K/2+a-1} &= (z)^{K/2+a-1}\left(\frac{K}{2}\widehat{\sigma}^2 + b^{-1}\right)^{1-(K/2+a)+K/2+a-1} \\ &= z^{K/2+a-1}. \end{aligned}$$

(Note that the supplemental material contains a typographical error, in that the exponent $K/2+a-1$

in Equation (3.12) should follow the second set of parentheses in the last term rather than the first set.)

Let

$$u = z\left(K\widetilde{\sigma}^2 + 2b^{-1}\right) = (n - k + 2a)\,\frac{\widetilde{\sigma}^2}{\sigma^2}$$

and

$$v = ab\widetilde{\sigma}^2.$$

Applying standard change of variables with the Jacobian equal to $(ab)^{-1}\left(Kv\,(ab)^{-1} + 2b^{-1}\right)^{-1} = (Kv + 2a)^{-1}$ gives the joint density

$$f\,(u, v) = \left(\frac{v}{ab}\right)^{K/2-1}\left(\frac{Kv}{2ab} + b^{-1}\right)^{1-(K/2+a)}\exp\left(-\frac{u}{2}\right)\left(\frac{u}{2}\right)^{(K+2a)/2-1}(Kv + 2a)^{-1}\,.$$

Separating the first term, factoring b^{-1} from the second term, and factoring $2a^{-1}$ from the last term gives

$$f\,(u, v) = \left(\frac{1}{ab}\right)^{K/2-1} v^{K/2-1}\exp\left(-\frac{u}{2}\right)\left(\frac{u}{2}\right)^{(K+2a)/2-1}\left[\left(\frac{Kv}{2a} + 1\right)b^{-1}\right]^{1-(K/2+a)}\left(\frac{Kv}{2a} + 1\right)^{-1}(2a)^{-1}\,.$$

(Note that the supplemental material contains a typographical error, as the denominator in the last term should be $2a$ rather than a.) After combining the second and third to the last terms,

$$f\,(u, v) = \left(\frac{1}{ab}\right)^{K/2-1}\left(b^{-1}\right)^{1-(K/2+a)}(2a)^{-1}\,v^{K/2-1}\exp\left(-\frac{u}{2}\right)\left(\frac{u}{2}\right)^{(K+2a)/2-1}\left(\frac{Kv}{2a} + 1\right)^{-(K/2+a)}\,.$$

Combining the a and b terms gives

$$f\,(u, v) = \frac{1}{2}\left(\frac{1}{a}\right)^{K/2} b^{-a}v^{K/2-1}\exp\left(-\frac{u}{2}\right)\left(\frac{u}{2}\right)^{(K+2a)/2-1}\left(\frac{Kv}{2a} + 1\right)^{-(K/2+a)}\,.$$

Eliminating the constant terms, this becomes

$$f(u,v) \propto (v)^{K/2-1} \left(\frac{K}{2a}v + 1\right)^{-(K/2+a)} \exp\left(-\frac{u}{2}\right)\left(\frac{u}{2}\right)^{(K+2a)/2-1}.$$

Using the definition of the F and χ^2 distributions, it can be seen that

$$f(u,v) \propto F_{K,2a}(v) \cdot \chi^2_{K+2a}(u). \tag{3.13}$$

(Note that the supplemental material contains a typographical error, giving the first distribution as $F_{2a,K}(v)$ rather than $F_{K,2a}(v)$.) The density in Equation (3.13) is easily factored into the two separate densities

$$f(u) \propto \chi^2_{K+2a}(u)$$

and

$$f(v) \propto F_{K,2a}(v).$$

Thus $(n-k+2a)\,\frac{\widetilde{\sigma}^2}{\sigma^2} \sim \chi^2_{n-k+2a}$ and $ab\left(\widetilde{\sigma}^2_s\right) \sim F_{n-k,2a}$. □

3.2 Implementation

Combining the results of Theorems 3.1 and 3.2, it can be seen that the RVM algorithm can be applied to a wide variety of GLMs by adjusting the residual sums of squares and the residual degrees of freedom. Furthermore, estimates of the hyperparameters a and b can be obtained from $\widehat{\sigma}^2$, the residuals from the standard GLM (without any assumptions on the distribution of the variances), using a numerical maximization routine.

The RVM algorithm does not constitute a specific software package, but an approach that may be applied to a variety of statistical problems. The RVM algorithm is easily applied to

class comparison problems for determining differential gene expression between experimental conditions. In the above formulae, k would equal the number of conditions, and $\boldsymbol{x}_m$ would be an indicator variable signifying the condition of chip m. With $k = 2$ conditions, the GLM reduces to the familiar t-test. For the two-sample comparison, let n_1 and n_2 be the sample sizes for conditions 1 and 2, $\widehat{\mu}_1$ and $\widehat{\mu}_2$ be the sample means, and $\widehat{\sigma}_1^2$ and $\widehat{\sigma}_2^2$ be the sample variances. Under the standard GLM (with no assumptions on $\widehat{\sigma}_1^2$ and $\widehat{\sigma}_2^2$), the pooled sample variance is

$$\widehat{\sigma}^2_{pooled} = \frac{(n_1 - 1)\,\widehat{\sigma}_1^2 + (n_2 - 1)\,\widehat{\sigma}_2^2}{n_1 + n_2 - 2}$$

and the test statistic is

$$t = \frac{\widehat{\mu}_1 - \widehat{\mu}_2}{\widehat{\sigma}_{pooled}\sqrt{1/n_1 + 1/n_2}} \sim t_{n-2}$$

Under RVM, the modified sample variance becomes

$$\widetilde{\sigma}^2_{pooled} = \frac{(n-2)\,\widehat{\sigma}^2_{pooled} + 2b^{-1}}{(n-2) + 2a}$$

and the modified test statistic is

$$\widetilde{t} = \frac{\widehat{\mu}_1 - \widehat{\mu}_2}{\widetilde{\sigma}_{pooled}\sqrt{1/n_1 + 1/n_2}} \sim t_{n-2+2a}$$

For comparing $k > 2$ conditions, the GLM uses the F-statistic

$$F = \frac{\left(\widehat{\widehat{SS}} - \widehat{SS}\right)/r}{\widehat{SS}\,/\,(n-k)} \sim F_{r,n-k}$$

Under RVM, this becomes the modified F-statistic given in Equation (3.3)

$$\widetilde{F} = \frac{\left(\widehat{\widehat{SS}} - \widehat{SS}\right)/r}{\widetilde{SS}\,/\,(n-k+2a)} \sim F_{r,n-k+2a}$$

Both the modified t and F-tests are implemented in the BRB-ArrayTools software package, which has been used extensively in the analysis of microarray data (see http://linus.nci.nih.gov/BRB-ArrayTools.html). The RVM framework extends in an analogous fashion to more complicated GLMs, although such formulations have not been presented in the original manuscript or subsequent publications.

3.3 Performance Assessment

Wright and Simon (2003) also provided initial simulation studies of the RVM algorithm, as well as applications to experimental datasets. The first simulated data set consisted of 6000 probesets from two groups with five samples in each group. This simulation design was repeated 2000 times. For each probeset s, the variance σ_s^2 was randomly generated from an inverse gamma distribution using the parameters $a = 3$ and $b = 1$. Intensities for each probeset were then randomly generated from a normal distribution with mean 0 and variance σ_s^2. For 3000 of the 6000 probesets, an amount between 0.1 and 2.0 was added to the intensities of the first group to represent differential expression between the two groups. A similar procedure was used to construct a simulated data set with 10 samples per group. Testing was then performed using a one-sided, two-sample t-test between groups. Three different methods were used for estimating the variance for the t-test: first, the pooled variance was used as the variance for all probesets; second, an individual variance was estimated separately for each gene; and third, the RVM algorithm was used to estimate variance. The first method produced an excessively high false-positive rate, while the last two had type I errors that were close to their expected values. However, the RVM algorithm had greater power for detecting a true difference in expression compared to the gene-level variance method, particularly for the smaller sample size of 5 arrays per group.

The performance of the RVM algorithm was also examined using actual data sets. Two data sets were used in this study: the DLBCL data set from Rosenwald *et al.* (2002), which consists of 7399 genes measured on 274 Lymphoma samples divided among three conditions; and the BRCA data set from Hedenfalk *et al.* (2001), which consists of 3226 genes measured on 22 breast cancer

samples divided among three conditions. For both data sets, the RVM estimates of the sample variances fitted the F distribution in Equation (3.11) well. Examination of the goodness-of-fit to the F distribution was chosen as it is impossible to obtain the true within group variances to determine if these follow the inverse gamma distribution given by the RVM algorithm. Subsequent analyses examining type I error and power were restricted to two conditions from the DLBCL data set. Given the large sample size, it was assumed that t-tests conducted using the full data for the two conditions could accurately identify differential expression. Genes with missing values of greater than 20% in the total data were excluded from analyses, leaving 6027 genes. The 1621 genes with $p < 0.0001$ were declared differentially expressed; the 2916 genes with $p > 0.05$ were declared nondifferentially expressed; and the 1490 genes with $0.05 > p > 0.001$ were declared indeterminate and not used in calculations of type I error or power. Next, 2000 repeated sub-samples of size 5 from each group were randomly chosen, and the p-value for the expression difference between conditions for each gene was computed. The number of significant p-values at different thresholds was compared to the number of significant p-values for the full data set to determine the type I error rate and power. A similar procedure was performed using sub-samples of size 10 from each group. Variances for the t-test of the subsamples were estimated using the pooled variance, the gene-level variance, and the RVM variance as in the simulation study. Again, the first method showed an excessively high false-positive rate, while the last two had rates that were similar to expected values. The RVM algorithm had greater power for detecting true differences in expression, especially for smaller sample sizes.

3.4 Limitations

Although the RVM method shows considerable promise for the analysis of gene expression and has been used in a number of experimental studies, there are limitations to address. The RVM method is a probeset-level analysis, rather than a probe-level analysis, so that there is a potential reduction in accuracy due to the use of expression summary values. The current implementation of the RVM method is unsuitable for analyzing probe-level data, as it assumes that error terms

in the model are independent of each other. Such an assumption is unrealistic for probes that are intended to target different regions of the same gene. The independence assumptions applied to probes within a probeset would lead to inflated variance estimates, as predicted by GLM theory, and reduced power for detecting differential expression. A more accurate model, which captures the correlation among the probes in a probeset, is necessary to overcome this problem.

A second potential limitation of the RVM method is the quality of the validation studies. The random variance model does appear to model the sample variances well, and the available simulation and experimental studies indicate that it has an acceptable level of accuracy in detecting differential gene expression. However, studies of the RVM method using spike-in data sets have not been conducted to date. Its accuracy has also not been compared to other class comparison methods beyond the pooled and individual variance t-tests. Further studies addressing this issue are warranted to establish clearly the relative merits of the RVM method.

Chapter 4

Extending Error Models

4.1 Introduction

Hypothesis testing for differential gene expression requires an estimate of the variance (or error) associated with the measurement of the expression values. Three fundamental approaches may be used to obtain such estimates. The first is a nonparametric approach, which assumes an underlying model for the variances exists but does not require the form of the model be specified. The second is a parametric approach that assumes the variances follow a given distribution that may be modeled, but does not impose a specific model on the intensities for the array. The third is a parametric approach that assumes a given distribution for the array intensities; based on the information from modeling the intensities, an estimate of the variances is also obtained. Each of these three approaches has its own advantages, and each is used in this research project.

4.2 Nonparametric Error Models

The nonparametric model for the variances associated with GeneChip intensity measurements assumes an underlying distribution of the variances exists that allows the variances to be computed, but does not require an explicit mathematical formulation of the distribution. The nonparametric approach is implicit in the original S-Score error model described by Equation (2.2), which assumes that probes with similar intensities would have similar variances. Thus, probes with similar intensities may be grouped together to obtain estimates for the variances without specific knowledge of the distribution of the variances.

A straightforward extension to the original S-Score error model is to assume that an individual probe will have the same expected intensity across multiple independent chips in the absence of differential expression, so that any variation would represent random rather than systematic error. Under this assumption, a measure of central tendency across chips can be used to estimate the true signal intensity. As the S-Score algorithm excludes outliers from its calculations, use of the mean as a measure of central tendency should be satisfactory. Convergence of the mean to the true signal value is assured under the Central Limit Theorem (CLT), although the rate of convergence varies depending on the nature of the underlying distribution. If the majority of probes are normally distributed in the absence of differential gene expression (Giles and Kipling, 2003), convergence would be rapid. Continuing the assumption of the original S-Score that probes with similar intensities also have similar variances, the error model in Equation (2.2) can be modified to accommodate more than two chips at a time.

This multichip S-Score groups the arrays in an experiment into two conditions 1 and 2, with each condition having multiple chips. Thus $N_c = 2$ as in the original S-Score, but N_m may be any positive number. The background noise estimates for condition c are defined as

$$\overline{bv}_1^2 = \frac{1}{N_1^2} \sum_{m=1}^{N_1} bv_{1m}^2 \qquad \text{and} \qquad \overline{bv}_2^2 = \frac{1}{N_2^2} \sum_{m=1}^{N_2} bv_{2m}^2$$

where bv_{cm} is the background estimate for the individual GeneChip in condition c. Similarly, the

signal intensities for probe pair p are defined as

$$\bar{y}_{1sp} = \frac{1}{N_1}\sum_{m=1}^{N_1} y_{1msp} \qquad \text{and} \qquad \bar{y}_{2sp} = \frac{1}{N_2}\sum_{m=1}^{N_2} y_{2msp}$$

where $y_{cmsp} = PM_{cmsp} - MM_{cmsp}$ is the $PM - MM$ difference for the individual Genechip in condition c. Using the proxy variance estimate from Equation (2.5),

$$\begin{aligned}
\text{var}\left(\bar{y}_{1sp}\right) &= \text{var}\left(\frac{1}{N_1}\sum_{m=1}^{N_1} y_{1msp}\right) \\
&= \frac{1}{N_1^2}\sum_{m=1}^{N_1} \text{var}\left(y_{1msp}\right) \\
&= \frac{1}{N_1^2}\sum_{m=1}^{N_1} y_{1msp}^2
\end{aligned}$$

and similarly for $\bar{y}_{2sp}$. These estimates are used in the computation of a revised error term analogous to Equation (2.2)

$$\varepsilon_{sp} = \sqrt{\gamma^2\left(\sum_{c=1}^{N_c}\left[\frac{1}{N_c^2}\sum_{m=1}^{N_m} y_{cmsp}^2\right]\right) + \sum_{c=1}^{N_c}\overline{bv}_c^2} \tag{4.1}$$

and a revised S-Score statistic analogous to Equation (2.4)

$$S_s = \sum_{p=1}^{N_p}\frac{\bar{y}_{2sp} - \bar{y}_{1sp}}{\alpha\varepsilon_{sp}\sqrt{N_p}}. \tag{4.2}$$

Under the assumptions of the original S-Score, the multichip S-Score statistic in Equation (4.2) is also normally distributed. With the appropriate choice of α, the distribution of the statistic would be the standard normal, allowing cutoff values recommended for the original S-Score to be used with the multichip version.

The distribution of the multichip S-Score remains empirically, rather than theoretically, based. However, the S-Score algorithm does have similarities to established results. If the number of chips in conditions 1 and 2 are equal, denoted as the number of chips in a condition N_m, then

Equation (4.2) may be rewritten as

$$
\begin{aligned}
S_s &= \frac{1}{\alpha\sqrt{N_p}} \sum_{p=1}^{N_p} \frac{\frac{1}{N_m}\sum_{m=1}^{N_m} y_{2msp} - \frac{1}{N_m}\sum_{m=1}^{N_m} y_{1msp}}{\sqrt{\gamma^2\left(\frac{1}{N_m^2}\sum_{m=1}^{N_m} y_{1msp}^2 + \frac{1}{N_m^2}\sum_{m=1}^{N_m} y_{2msp}^2\right) + \frac{1}{N_m^2}\sum_{m=1}^{N_m} b_{1m}^2 + \frac{1}{N_m^2}\sum_{m=1}^{N_m} b_{2m}^2}} \\
&= \frac{1}{\alpha\sqrt{N_p}} \sum_{p=1}^{N_p} \frac{\sum_{m=1}^{N_m} y_{2msp} - \sum_{m=1}^{N_m} y_{1msp}}{\sqrt{\gamma^2\left(\sum_{m=1}^{N_m} y_{1msp}^2 + \sum_{m=1}^{N_m} y_{2msp}^2\right) + \sum_{m=1}^{N_m} b_{1m}^2 + \sum_{m=1}^{N_m} b_{2m}^2}} \\
&= \frac{1}{\alpha\sqrt{N_p}} \sum_{p=1}^{N_p} \frac{\sum_{m=1}^{N_m} \left(y_{2msp} - y_{1msp}\right)}{\sqrt{\sum_{m=1}^{N_m} \left[\gamma^2\left(y_{1msp}^2 + y_{2msp}^2\right) + b_{1m}^2 + b_{2m}^2\right]}}.
\end{aligned}
\tag{4.3}
$$

Sums in the form of the summand in Equation (4.3) can generally be shown to follow a standard normal distribution, provided that the variance of the individual random variables is finite and no single variance term dominates the sum. Such generalizations of the CLT are discussed in depth in Billingsley (1986, chapter 27) and Resnick (1999, Section 9.8), which also describe specific conditions necessary for the theorems to hold. Unfortunately, these results do not constitute a proof of the distribution for the S-Score statistic, as the proxy in Equation (2.5) has not been established as a proper estimator for the variance of the intensities. However, these results do suggest that such a proof may be possible, though the variance estimator may need to be altered.

One of the central problems in estimating the probe-level variance in Equation (2.2) is that there is only one observation for each probe on a chip, so that the variance cannot be computed with the usual formulae. Jain *et al.* (2003) have proposed an method of estimating probeset-level variances with small numbers of replicates, which they call the local pooled error (LPE), that may be readily adapted from a probeset-level to a probe-level model. The LPE method assumes that probesets with similar intensities will have similar variances, in keeping with the intensity-based error model used in the S-Score. It is based on information from the MA plot, a standard plot for

examining patterns of intensity variation across chips (Dudoit *et al.*, 2002). For a pair of chips, the MA plot depicts the difference between the log intensities (designated as M) versus the average of the log intensities (designated as A). With two observations, the difference is proportional to the square root of the variance, so the MA plot provides information about the variability of the chips as a function of mean intensity (Jain *et al.*, 2003). The LPE algorithm constructs an error distribution function for the probeset intensities using a nonparametric local regression to fit the differences versus the averages.

The LPE method is described in Algorithm 4.1. Specific points of this algorithm deserve further discussion. First, assuming no differential expression exists within a condition, the expected value of the differences $\mathcal{E}(lm) = 0$, but the mean of the differences using only distinct pairs of arrays may differ significantly from 0 due to stochastic error. The LPE method includes all pairwise comparisons in the calculation of lm and $\bar{y}$, which guarantees that the mean of lm will be zero. However, it requires that the variances be adjusted by a constant factor to account for this duplication. Second, it is possible to have variance estimates with a small negative value after fitting the nonparametric regression, particularly for the lower intensity range where many probesets may have similar intensity values. The LPE method corrects for this by imputing the lowest observed variance to any negative values, a reasonable though admittedly arbitrary practice. Finally, the upper tail of the average intensities $\bar{y}$ is are sparsely populated. As the quantiles contain approximately the same number of observations, the upper quantiles contain a large range of values. The variance within these quantiles may be spuriously high, as intensities over such a large range would be expected to have dissimilar variances. To compensate for the excessively large estimates of variances for the upper tail, Jain *et al.* (2005) added adaptive intervals to the LPE method, which is described in Algorithm 4.2. Under the Adaptive Interval (AI) algorithm, an initial error distribution is obtained by the LPE method with the average intensities grouped by quantiles. This initial error distribution is used to compute the estimated variance for each gene based on the median intensity across all chips within a condition. New groups of the average intensities are then constructed so that all values in the group are within one standard deviation of the smallest value in the group; more values may be included in the group to ensure that each

Algorithm 4.1 Local Pooled Error Algorithm

1: Compute differences lm and averages $\overline{y}$ of the probeset intensities y_{cms} for all pairs of arrays (m_1, m_2) in condition c
$lm_{cm_1m_2s} = \log_2 \frac{y_{cm_1s}}{y_{cm_2s}} = \log_2 y_{cm_1s} - \log_2 y_{cm_2s}$
$\overline{y}_{cm_1m_2s} = \log_2 \sqrt{y_{cm_1s} y_{cm_2s}} = \frac{1}{2}(\log_2 y_{cm_1s} + \log_2 y_{cm_2s})$
2: Divide the averages $\overline{y}$ for condition c into N_Q quantiles $Q_1, Q_2, \ldots, Q_{N_Q}$, with default $N_Q = 100$
3: Pool the differences lm and averages $\overline{y}$ into $N_Q - 1$ groups $O_{co}, o = 1, 2, \ldots, N_Q - 1$, based on quantiles
$\overline{y}_{cm_1m_2s} \in O^{\overline{y}}_{cq}, lm_{cm_1m_2s} \in O^{lm}_{cq}$ if $Q_o < \overline{y}_{cm_1m_2s} \leq Q_{o+1}$
4: Compute the median and variance for each group
$\xi_{co} = \text{median}\left(O^{\overline{y}}_{co}\right)$
$\sigma^2_{co} = \text{var}\left(O^{lm}_{co}\right)$
5: Obtain the error distribution for condition c by fitting a nonparametric local regression (smoothing spline) for the variances on the medians
6: Impute the lowest observed variance, $\min\left(\sigma^2_{c1}, \sigma^2_{c2}, \ldots, \sigma^2_{c,N_Q}\right)$, to negative variance estimates

group has a prespecified minimum size. These new groups based on the adaptive intervals are used to obtain a revised error distribution, and the revised error distribution is used in testing for differential gene expression.

Algorithm 4.2 Adaptive Interval Algorithm

1: Estimate the baseline variance function for all arrays in condition c using the LPE algorithm
2: Compute the median intensity ξ_{cs} for each probeset in condition c
3: Using the baseline variance function, compute variance estimates σ_{cs} for each probeset in condition c
4: Order the median intensities and variances by the median intensity, with the the ordered medians and variances denoted as $\xi_{c(s)}$ and $\sigma_{c(s)}$ respectively
5: Set the lower and upper threshold values for the first interval to $\xi_{c(1)}$ and $\xi_{c(1)} + \sigma_{c(1)}$
6: Set the lower and upper threshold values for the next interval to $\xi_{c(i)}$ and $\xi_{c(i)} + \sigma_{c(i)}$, where $i = \min\left(j : \xi_{c(j)} \geq \xi_{c(j-1)} + \sigma_{c(j-1)}\right)$
7: Repeat step 6 until all the data are assigned to intervals
8: Compute the adaptive interval variance function for all arrays in condition c using the LPE algorithm with the new adaptive intervals

As the LPE method is based on similar assumptions regarding the error distribution for microarray intensities as the S-Score, features of the former may be incorporated into the latter for a more mathematically rigorous estimate of the error term. This Pooled S-Score algorithm is described in Algorithm 4.3. This algorithm does depart from the LPE method in several ways.

First, rather than computing all pairwise means and differences, the mean intensity is calculated across each probe for all chips within a condition. This would lead to more accurate estimates of the average intensity for each probe, but would also lead to fewer observations for computing the error distribution than using pairwise estimates. However, given that the number of probes on an array is roughly an order of magnitude greater than the number of probesets, averaging probe intensities across all chips should still result in sufficient data for fitting the error distribution. Second, a parametric quadratic function is used in fitting the average intensities to the variances. This avoids the possibility of negative variance estimates that can occur with the nonparametric smoothing spline. It is also in keeping with the original S-Score error model and with the empiric distribution seen in plots of the average intensities for a probe versus the variance of the probe across chips. Finally, because of the arbitrary nature of some decisions in the AI algorithm, adaptive intervals were not used in the Pooled S-Score. Instead, a robust regression was used in fitting the error model to minimize the influence of outliers.

Algorithm 4.3 Pooled S-Score Error Algorithm

1: Compute averages $\overline{y}$ of the $PM - MM$ probe intensities y_{cmsp} for all arrays in condition c
$\overline{y}_{csp} = \frac{1}{N_m}\left(y_{c1sp} + y_{c2sp} + \ldots + y_{c,N_m,sp}\right)$
2: Divide the averages $\overline{y}$ for condition c into Q quantiles $q_1, q_2, \ldots, q_Q$, with default $Q = 1000$
3: Pool the intensities y into O groups $O_{co}, g = 1, 2, \ldots, N_Q - 1$, based on quantiles
$y_{c1sp}, y_{c2sp}, \ldots, y_{c,N_m,sp} \in O^y_{co}$ if $Q_o < \overline{y}_{csp} \leq Q_{o+1}$
4: Compute the 25% trimmed mean and median absolute deviation for each group
$\mu_{co} = \text{mean}\,(O^y_{co}, 0.25)$
$\sigma^2_{co} = \text{MAD}\,(O^y_{co})$
5: Obtain the error distribution for condition c by fitting robust quadratic regression for the MAD on the squared trimmed means
6: Compute the error term ε for each probe and use this error term in the calculation of the S-Score statistic

4.3 Parametric Error Models for Variances Only

A second type of multivariate error model would be a parametric model similar to the univariate RVM. This approach requires a distributional assumption, which is made only for the variances of the probe-level intensities rather than the means and the variances. A multivariate RVM would incorporate the strength of the univariate RVM by borrowing information across probesets, which would lead to greater power to detect gene expression differences. However, modeling information at the probe level requires several results from multivariate statistical theory to capture the correlation among probes in a probeset.

4.3.1 Prerequisite Matrix Algebra

Multivariate statistical theory relies heavily on linear and matrix algebra. The topic of matrix algebra relevant to statistics is covered by several authors, including Searle (1982), Graybill (1983), and Harville (1997). Although much of the material in these reference works is familiar to statisticians, some less commonly known results are needed in the development of the multivariate RVM model. The next section reviews these results, followed by proofs generalizing the RVM method to the multivariate case.

4.3.1.1 Basic Results and Definitions

Definition 4.1. ***Symmetric Matrix.*** *The $p \times p$ matrix $\boldsymbol{X}$ is said to be symmetric if $\boldsymbol{X} = \boldsymbol{X}'$.*

Several important properties of symmetric matrices have been noted, including

1. If $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are $p \times p$ symmetric matrices, then $\boldsymbol{X}_1 + \boldsymbol{X}_2$ is symmetric.

2. If $\boldsymbol{X}_1$ is a $p \times p$ symmetric matrix and $\boldsymbol{X}_2$ is any $p \times p$ matrix, then $\boldsymbol{X}_2'\boldsymbol{X}_1\boldsymbol{X}_2$ and $\boldsymbol{X}_2\boldsymbol{X}_1\boldsymbol{X}_2'$ are symmetric.

3. If $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are $p \times p$ symmetric matrices, the product $\boldsymbol{X}_1\boldsymbol{X}_2$ is generally *not* symmetric.

4. If $\boldsymbol{X}$ is any $p_1 \times p_2$ matrix, then $\boldsymbol{X}'\boldsymbol{X}$ and $\boldsymbol{X}\boldsymbol{X}'$ are symmetric.

Definition 4.2. ***Positive Definite Matrix.*** *The $p \times p$ symmetric matrix $\boldsymbol{X}$ is said to be positive definite if $\boldsymbol{y'Xy} > 0$ for all possible vectors $\boldsymbol{y} \neq \boldsymbol{0}$.*

Definition 4.3. ***Positive Semidefinite Matrix.*** *The $p \times p$ symmetric matrix $\boldsymbol{X}$ is said to be positive semidefinite if $\boldsymbol{y'Xy} \geq 0$ for all possible vectors $\boldsymbol{y} \neq \boldsymbol{0}$.*

Definition 4.4. ***Nonnegative Definite Matrix.*** *The $p \times p$ symmetric matrix $\boldsymbol{X}$ is said to be nonnegative definite if $\boldsymbol{X}$ is either positive definite or positive semidefinite.*

Several important properties of positive definite and positive semidefinite matrices have been described, including

1. If $\boldsymbol{X}$ is nonnegative definite, then $\boldsymbol{X}$ is nonsingular if and only if $\boldsymbol{X}$ is positive definite.

2. If $\boldsymbol{X}_1$ is a $p \times p$ positive definite matrix and $\boldsymbol{X}_2$ is a $p \times p$ nonnegative definite matrix, then $\boldsymbol{X}_1 + \boldsymbol{X}_2$ is a positive definite matrix.

3. If $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are $p \times p$ positive semidefinite matrices, then $\boldsymbol{X}_1 + \boldsymbol{X}_2$ is positive semidefinite.

4. If $\boldsymbol{X}_1$ is a $p \times p$ positive definite matrix and $\boldsymbol{X}_2$ is a $p \times p$ matrix, then $\boldsymbol{X}_2\boldsymbol{X}_1\boldsymbol{X}_2'$ is positive definite if $\text{rank}\,(\boldsymbol{X}_2) = p$ and positive semidefinite if $\text{rank}\,(\boldsymbol{X}_2) < p$.

5. If $\boldsymbol{X}$ is any $p \times p$ matrix, then $\boldsymbol{XX'}$ is positive definite if $\boldsymbol{X}$ is nonsingular and positive semidefinite if $\boldsymbol{X}$ is singular.

Definition 4.5. ***Trace of a Matrix.*** *Let $\boldsymbol{X}$ be a $p \times p$ matrix. The trace of $\boldsymbol{X}$, denoted as $\text{Tr}\,(\boldsymbol{X})$, is defined as*

$$\text{Tr}\,(\boldsymbol{X}) = \sum_{i=1}^{p} x_{ii}$$

where $x_{ii}, i = 1, 2, \ldots, p$ are the diagonal elements of $\boldsymbol{X}$.

The trace provides a scalar summary of a square matrix. Several important properties of the trace include

1. For any square matrix $\boldsymbol{X}$, $\mathrm{Tr}\,(\boldsymbol{X}) = \mathrm{Tr}\,(\boldsymbol{X}')$.

2. For any two square matrices $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, $\mathrm{Tr}\,(\boldsymbol{X}_1 + \boldsymbol{X}_2) = \mathrm{Tr}\,(\boldsymbol{X}_1) + \mathrm{Tr}\,(\boldsymbol{X}_2)$.

3. For any square matrix $\boldsymbol{X}$ and any scalar constant φ, $\mathrm{Tr}\,(\varphi\boldsymbol{X}) = \varphi\,\mathrm{Tr}\,(\boldsymbol{X})$.

4. For any three square matrices $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, and $\boldsymbol{X}_3$, $\mathrm{Tr}\,(\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_3) = \mathrm{Tr}\,(\boldsymbol{X}_2\boldsymbol{X}_3\boldsymbol{X}_1) = \mathrm{Tr}\,(\boldsymbol{X}_3\boldsymbol{X}_1\boldsymbol{X}_2)$.

Definition 4.6. ***Determinant of a Matrix.*** *Let $\boldsymbol{X}$ be a $p \times p$ matrix. The determinant of $\boldsymbol{X}$, denoted as $|\boldsymbol{X}|$ or $\det(\boldsymbol{X})$, is defined as*

$$|\boldsymbol{X}| = \sum_{(j_1, j_2, \ldots, j_p)} (-1)^{\varphi(j_1, j_2, \ldots, j_p)} \prod_{i=1}^{p} x_{ij_i} \tag{4.4}$$

where the permutation function $\varphi(\cdot)$ denotes a permutation $j_1, j_2, \ldots, j_p$ of the integers $1, 2, \ldots, p$ and the summation is taken over all distinct permutations.

The determinant provides a scalar summary of a square matrix. There are also other ways of defining the determinant. The most relevant to the present work is the following: Let $\boldsymbol{X}_{-i,-j}$ denote the $(p-1) \times (p-1)$ submatrix of $\boldsymbol{X}$ by deleting the ith row and the jth column from $\boldsymbol{X}$. The cofactor cf_{ij} of element x_{ij} is the quantity

$$cf_{ij} = (-1)^{(i+j)} \left|\boldsymbol{X}_{-i,-j}\right|$$

and $\left|x_{ij}\right| = x_{ij}$. The matrix $\boldsymbol{X}^{\#\prime} = \left(cf_{ij}\right)$ is called the *cofactor matrix* of $\boldsymbol{X}$, and the matrix $\boldsymbol{X}^{\#}$ or $\mathrm{adj}\,(\boldsymbol{X})$ is called the *adjoint* of $\boldsymbol{X}$. Using this recursive definition of the cofactor, the determinant can be expressed as

$$|\boldsymbol{X}| = \sum_{j=1}^{p} cf_{ij} x_{ij} \quad (i = 1, 2, \ldots, p)\,. \tag{4.5}$$

Several important properties of the determinant include

1. For any square matrix $\boldsymbol{X}$, $|\boldsymbol{X}| = |\boldsymbol{X}'|$.

2. For any two square matrices $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, $|\boldsymbol{X}_1\boldsymbol{X}_2| = |\boldsymbol{X}_1|\,|\boldsymbol{X}_2|$.

3. For any $p \times p$ square matrix $\boldsymbol{X}$ and any scalar constant ζ, $|\zeta\boldsymbol{X}| = \zeta^p\,|\boldsymbol{X}|$.

4. For any square matrix $\boldsymbol{X}$, $\boldsymbol{X}^{\#}\boldsymbol{X} = |\boldsymbol{X}|\,\boldsymbol{I}_p$

4.3.1.2 Matrix Decompositions

Definition 4.7. ***Eigenvalues and Eigenvectors.*** *Let $\boldsymbol{X}$ be any $p \times p$ matrix. Then the scalar λ is defined to be an eigenvalue of $\boldsymbol{X}$ if there exists a $p \times 1$ vector $\boldsymbol{e} \neq \boldsymbol{0}$ such that $\boldsymbol{X}\boldsymbol{e} = \lambda\boldsymbol{e}$. The vector $\boldsymbol{e}$ is defined to be the eigenvector of $\boldsymbol{X}$ corresponding to λ.*

There are several salient properties of eigenvalues and eigenvectors relevant to the present work, including

1. $\sum_{i=1}^{p} \lambda_i = \mathrm{Tr}\,(\boldsymbol{X})$, where $\lambda_1, \lambda_2, \ldots, \lambda_p$ are the eigenvalues of $\boldsymbol{X}$.

2. $\prod_{i=1}^{p} \lambda_i = |\boldsymbol{X}|$, where $\lambda_1, \lambda_2, \ldots, \lambda_p$ are the eigenvalues of $\boldsymbol{X}$.

3. If $\boldsymbol{X}_1$ is a $p_1 \times p_2$ matrix and $\boldsymbol{X}_2$ is a $p_2 \times p_1$ matrix, then the nonzero eigenvalues of $\boldsymbol{X}_1\boldsymbol{X}_2$ are the same as the nonzero eigenvalues of $\boldsymbol{X}_2\boldsymbol{X}_1$.

Definition 4.8. ***Spectral Decomposition.*** *Let $\boldsymbol{X}$ be a $p \times p$ symmetric matrix. The spectral decomposition of $\boldsymbol{X}$ is defined as $\boldsymbol{X} = \boldsymbol{E}\boldsymbol{L}\boldsymbol{E}'$, where $\boldsymbol{L} = \mathrm{diag}\left(\lambda_1, \lambda_2, \ldots, \lambda_p\right)$; $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ are the ordered eigenvalues of $\boldsymbol{X}$; and $\boldsymbol{E}$ is the matrix of normalized eigenvectors corresponding to the eigenvalues.*

Note that the matrix $\boldsymbol{E}$ is orthogonal, so that $\boldsymbol{E}'\boldsymbol{E} = \boldsymbol{I}_p$. The spectral decomposition is only defined for square symmetric matrices.

Definition 4.9. ***Square Root of a Positive Definite Matrix.*** *Let $\boldsymbol{X}$ be a $p \times p$ nonnegative definite matrix. Let $\boldsymbol{X} = \boldsymbol{E}\boldsymbol{L}\boldsymbol{E}'$ be the spectral decomposition of $\boldsymbol{X}$, where $\boldsymbol{L} = \mathrm{diag}\left(\lambda_1, \lambda_2, \ldots, \lambda_p\right)$ with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ being the ordered eigenvalues of $\boldsymbol{X}$. The square root of the matrix $\boldsymbol{X}$, denoted*

as $\boldsymbol{X}^{1/2}$*, will be defined as*

$$\boldsymbol{X}^{1/2} = \boldsymbol{E}\begin{bmatrix} \sqrt{\lambda_1} & 0 & 0 & \dots & 0 \\ 0 & \sqrt{\lambda_2} & 0 & \dots & 0 \\ 0 & 0 & \sqrt{\lambda_3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sqrt{\lambda_p} \end{bmatrix}\boldsymbol{E}', \tag{4.6}$$

so that $\boldsymbol{X}^{1/2}\boldsymbol{X}^{1/2} = \boldsymbol{X}$.

Other definitions of the square root of a matrix exist, for which the only requirement of a square root matrix $\boldsymbol{X}^*$ is that $\boldsymbol{X}^*\boldsymbol{X}^{*\prime} = \boldsymbol{X}$ or $\boldsymbol{X}^{*\prime}\boldsymbol{X}^* = X$ (Olkin and Rubin, 1964; Rencher, 2002, p. 25). However, with such alternative definitions, the matrix $\boldsymbol{X}^*$ is not necessarily unique, while the matrix $\boldsymbol{X}^{1/2}$ given in Equation (4.6) is unique. The matrix $\boldsymbol{X}^{1/2}$ is occasionally called the *positive definite* square root of a matrix (Muirhead, 1982, p. 588) to distinguish it from the alternative definitions, but will simply be referred to as the square root throughout this work. Note that $\boldsymbol{X}^{1/2}$ is symmetric, implying that $\boldsymbol{X}^{1/2}\boldsymbol{X}^{1/2\prime} = \boldsymbol{X}^{1/2\prime}\boldsymbol{X}^{1/2} = \boldsymbol{X}$, so $\boldsymbol{X}^{1/2}$ may often be used as the square root when alternative definitions have been used. Furthermore, note that $\boldsymbol{X}^{1/2}$ is positive definite if $\boldsymbol{X}$ is positive definite. Then the inverse of $\boldsymbol{X}^{1/2}$, denoted $\boldsymbol{X}^{-1/2}$, exists and satisfies the relationship $\boldsymbol{X}^{-1/2}\boldsymbol{X}^{-1/2} = \boldsymbol{X}^{-1}$. If $\boldsymbol{X}$ is positive semidefinite, then $\boldsymbol{X}^{1/2}$ is also positive semidefinite.

4.3.1.3 Kronecker Product

This section reviews some key results concerning the Kronecker product, which has also been called the direct product. The Kronecker product arises naturally in the development of matrix derivatives and differentials as described in later sections. A comprehensive review is provided by Graham (1981), while a discussion of statistical applications is given by Neudecker (1968), Neudecker (1969), and Koning *et al.* (1991).

Definition 4.10. ***Kronecker Product.*** *The Kronecker product of the* $p_1 \times p_2$ *matrix* $\boldsymbol{X}_1$ *and the*

$p_3 \times p_4$ matrix $\boldsymbol{X}_2$ with elements

$$\boldsymbol{X}_1 = \begin{bmatrix} x_{111} & x_{121} & x_{131} & \dots & x_{1p_21} \\ x_{211} & x_{221} & x_{231} & \dots & x_{2p_21} \\ x_{311} & x_{321} & x_{331} & \dots & x_{3p_21} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{p_11} & x_{p_121} & x_{p_131} & \dots & x_{p_1p_21} \end{bmatrix} \quad \textit{and } \boldsymbol{X}_2 = \begin{bmatrix} x_{112} & x_{122} & x_{132} & \dots & x_{1p_42} \\ x_{212} & x_{222} & x_{232} & \dots & x_{2p_42} \\ x_{312} & x_{322} & x_{332} & \dots & x_{3p_42} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{p_312} & x_{p_322} & x_{p_332} & \dots & x_{p_3p_42} \end{bmatrix}$$

is denoted $\boldsymbol{X}_1 \otimes \boldsymbol{X}_2$ and is defined to be the $p_1p_3 \times p_2p_4$ matrix

$$\boldsymbol{X}_1 \otimes \boldsymbol{X}_2 = \begin{bmatrix} x_{111}\boldsymbol{X}_2 & x_{121}\boldsymbol{X}_2 & x_{131}\boldsymbol{X}_2 & \dots & x_{1p_21}\boldsymbol{X}_2 \\ x_{211}\boldsymbol{X}_2 & x_{221}\boldsymbol{X}_2 & x_{231}\boldsymbol{X}_2 & \dots & x_{2p_21}\boldsymbol{X}_2 \\ x_{311}\boldsymbol{X}_2 & x_{321}\boldsymbol{X}_2 & x_{331}\boldsymbol{X}_2 & \dots & x_{3p_21}\boldsymbol{X}_2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{p_11}\boldsymbol{X}_2 & x_{p_121}\boldsymbol{X}_2 & x_{p_131}\boldsymbol{X}_2 & \dots & x_{p_1p_21}\boldsymbol{X}_2 \end{bmatrix}$$

Note that, unlike the usual matrix product $\boldsymbol{X}_1\boldsymbol{X}_2$, the Kronecker product is defined regardless of the dimension of $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$. The Kronecker product $\boldsymbol{X}_1 \otimes \boldsymbol{X}_2$ will be of different dimension than $\boldsymbol{X}_2 \otimes \boldsymbol{X}_1$ unless $p_1 = p_3$ and $p_2 = p_4$. Even in the latter case, $\boldsymbol{X}_1 \otimes \boldsymbol{X}_2 \neq \boldsymbol{X}_2 \otimes \boldsymbol{X}_1$ generally. Other well-known properties of the Kronecker product (Graham, 1981, section 2.3) include

1. For any scalar constant c and any two matrices $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, $(c\boldsymbol{X}_1) \otimes \boldsymbol{X}_2 = \boldsymbol{X}_1 \otimes (c\boldsymbol{X}_2) = c\,(\boldsymbol{X}_1 \otimes \boldsymbol{X}_2)$.

2. $\boldsymbol{X}_1 \otimes (\boldsymbol{X}_2 + \boldsymbol{X}_3) = (\boldsymbol{X}_1 \otimes \boldsymbol{X}_2) + (\boldsymbol{X}_1 \otimes \boldsymbol{X}_3)$ for any three matrices for which the matrix sum is defined.

3. $(\boldsymbol{X}_1 \otimes \boldsymbol{X}_2)\,(\boldsymbol{X}_3 \otimes \boldsymbol{X}_4) = \boldsymbol{X}_1\boldsymbol{X}_3 \otimes \boldsymbol{X}_2\boldsymbol{X}_4$ for any four matrices for which the matrix product is defined.

4.3.1.4 Vec and Vech Operators

This section reviews the vec and vech operators, which transform a matrix into a column vector. Interest in such a transformation has existed for some time; Neudecker (1969) and Henderson and Searle (1979) show the usefulness of the vec operator in the computation of matrix derivatives and Jacobians, as described below. The vec operator also facilitates the expression of a multivariate model as a corresponding univariate model, as developed by Searle (1978).

Definition 4.11. ***Vec Operator.*** *For the $p_1 \times p_2$ matrix $\boldsymbol{X}$, the* vec *operator results in the stacking of the columns of $\boldsymbol{X}$ into a $p_1 p_2 \times 1$ column vector. Thus, for the matrix*

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1p_2} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2p_2} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3p_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{p_1 1} & x_{p_1 2} & x_{p_1 3} & \dots & x_{p_1 p_2} \end{bmatrix},$$

the application of the vec *operator to $\boldsymbol{X}$, denoted as* vec $(\boldsymbol{X})$*, is defined as*

$$\text{vec}\,(\boldsymbol{X}) = \begin{bmatrix} x_{11} \\ x_{21} \\ \vdots \\ x_{p_1 1} \\ x_{12} \\ x_{22} \\ \vdots \\ x_{p_1 2} \\ \vdots \\ x_{p_1 p_2} \end{bmatrix}$$

The vec operator is related to the Kronecker product through the identity

$$\operatorname{vec}(\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_3) = (\boldsymbol{X}_3' \otimes \boldsymbol{X}_1)\operatorname{vec}\boldsymbol{X}_2 \tag{4.7}$$

for any three matrices $\boldsymbol{X}_1$, $\boldsymbol{X}_2$, and $\boldsymbol{X}_3$ such that the product is defined. Also,

$$\boldsymbol{v}_1 \otimes \boldsymbol{v}_2 = \operatorname{vec}(\boldsymbol{v}_2\boldsymbol{v}_1') \tag{4.8}$$

for any two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$. The vec operator is also related to the trace function through the identity

$$\operatorname{Tr}(\boldsymbol{X}_1'\boldsymbol{X}_2) = (\operatorname{vec}\boldsymbol{X}_1)'\operatorname{vec}\boldsymbol{X}_2. \tag{4.9}$$

If $\boldsymbol{X}$ has no specific pattern, then $\operatorname{vec}(\boldsymbol{X})$ contains all of the elements of $\boldsymbol{X}$ arranged in a single vector. However, if $\boldsymbol{X}$ is a $p \times p$ symmetric matrix, it contains only $\frac{1}{2}p(p+1)$ unique elements, as the remainder may be deduced from symmetry constraints. An operator analogous to the vec operator is necessary to capture only the unique elements in the column vector.

Definition 4.12. ***Vech Operator.*** *For the $p \times p$ symmetric matrix $\boldsymbol{X}$, the* vech *operator results in the stacking of the elements of each column of $\boldsymbol{X}$ that are on or below the diagonal into a $\frac{1}{2}p(p+1) \times 1$ column vector. Thus, for the matrix*

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1p_2} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2p_2} \\ x_{31} & x_{32} & x_{33} & \dots & x_{3p_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{p_1 1} & x_{p_1 2} & x_{p_1 3} & \dots & x_{p_1 p_2} \end{bmatrix},$$

where $x_{ij} = x_{ji}$ for all $i \neq j$, the application of the vech *operator to X, denoted as* $\operatorname{vech}(\boldsymbol{X})$*, is*

defined as

$$\operatorname{vech}(\boldsymbol{X}) = \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \\ \vdots \\ x_{p_1 1} \\ x_{22} \\ x_{32} \\ \vdots \\ x_{p_1 2} \\ x_{33} \\ x_{43} \\ \vdots \\ x_{p_1 3} \\ \vdots \\ x_{p_1 p_2} \end{bmatrix}$$

Henderson and Searle (1979) introduce the vech operator and describe the its use in the computation of matrix derivatives and Jacobians of symmetric matrices, in a manner analogous to the vec operator. Magnus and Neudecker (1980) describe a similar operator, which they denote as $\upsilon(X)$. In later publications, they adopted the use of the vech operator (Abadir and Magnus, 2005), which will be the notation used throughout this work.

It should be noted that, in general, $\operatorname{vec}(\boldsymbol{X}_1) = \operatorname{vec}(\boldsymbol{X}_2) \nRightarrow \boldsymbol{X}_1 = \boldsymbol{X}_2$, as $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ may not be of the same dimension. However, if $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are of the same dimension, then $\operatorname{vec}(\boldsymbol{X}_1) = \operatorname{vec}(\boldsymbol{X}_2) \Rightarrow \boldsymbol{X}_1 = \boldsymbol{X}_2$. Since the operand of the vech operator is square, it follows that $\operatorname{vech}(\boldsymbol{X}_1) = \operatorname{vech}(\boldsymbol{X}_2) \Rightarrow \boldsymbol{X}_1 = \boldsymbol{X}_2$ for symmetric $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$. However, if $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ are not symmetric, then $\operatorname{vech}(\boldsymbol{X}_1) = \operatorname{vech}(\boldsymbol{X}_2) \nRightarrow \boldsymbol{X}_1 = \boldsymbol{X}_2$, as the supradiagonal elements of the two matrices may not be

the same.

4.3.1.5 Patterned Matrices: Elimination, Duplication, Commutation, and Symmetrizer Matrices

This section reviews the development and properties of certain patterned matrices that perform transformations on the vec and vech operators. In addition, each matrix has related properties that make them useful in the computation of certain Kronecker products. These patterned matrices are introduced by Magnus and Neudecker (1979, 1980), with additional results given by Magnus (1988). Henderson and Searle (1979) also developed a similar system of patterned matrices at approximately the same time, which was extended in Searle (1982, section 12.9). The former notation will be used throughout this work; correspondence with the latter notation will be given in this subsection only.

If the $p \times p$ matrix $\boldsymbol{X}$ is symmetric, it is obvious that $\operatorname{vec}(\boldsymbol{X})$ contains the same elements as $\operatorname{vech}(\boldsymbol{X})$, with some of these elements being repeated. A similar observation may be noted for lower and upper triangular matrices, as the supradiagonal and infradiagonal elements respectively are 0. Thus, the vec and vech operators are linear transformations of each other in such circumstances. Magnus and Neudecker (1980, 1999) define the *elimination matrix*, denoted by $\boldsymbol{L}_p$, as the $\frac{1}{2}p(p+1) \times p^2$ patterned matrix that performs the transformation

$$\boldsymbol{L}_p \operatorname{vec}(\boldsymbol{X}) = \operatorname{vech}(\boldsymbol{X}) \tag{4.10}$$

for any $p \times p$ matrix $\boldsymbol{X}$. In other words, the matrix $\boldsymbol{L}_p$ eliminates the supradiagonal elements from the matrix $\boldsymbol{X}$. Henderson and Searle (1979) describe a similar matrix, which they denote with $\boldsymbol{H}$. Although the two are equivalent for symmetric matrices, the definition by Magnus and Neudecker (1980) is more versatile as it applies to triangular matrices as well.

The elimination matrix $\boldsymbol{L}_p$ is unique and of full row rank. The central property of $\boldsymbol{L}_p$ as applied to the vec operator is that $\boldsymbol{L}_p'\boldsymbol{L}_p \operatorname{vec}(\boldsymbol{X}) = \operatorname{vec}(\boldsymbol{X})$ for any lower triangular matrix $\boldsymbol{X}$. The corresponding Kronecker property for the elimination matrix is that $\boldsymbol{L}'\boldsymbol{L}_p \left(\boldsymbol{X}_1' \otimes \boldsymbol{X}_2\right) \boldsymbol{L}_p' =$

$\left(\boldsymbol{X}_1' \otimes \boldsymbol{X}_2\right) \boldsymbol{L}_p'$ for any two lower triangular matrices $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$.

Magnus and Neudecker (1980, 1999) define the *duplication matrix*, denoted by $\boldsymbol{D}_p$, as the $p^2 \times \frac{1}{2}p\,(p+1)$ matrix that performs the transformation

$$\boldsymbol{D}_p \operatorname{vech}(\boldsymbol{X}) = \operatorname{vec}\left(\boldsymbol{X} + \boldsymbol{X}' - \operatorname{diag}(\boldsymbol{X})\right). \tag{4.11}$$

Thus, the duplication matrix duplicates the infradiagonal elements of a matrix to the supradiagonal elements, transforming $\operatorname{vech}(\boldsymbol{X})$ into the vec of a symmetric matrix. If $\boldsymbol{X}$ is a symmetric matrix, then

$$\boldsymbol{D}_p \operatorname{vech}(\boldsymbol{X}) = \operatorname{vec}(\boldsymbol{X}). \tag{4.12}$$

Henderson and Searle (1979) describe a similar matrix, denoted as $\boldsymbol{G}$, though this matrix is only applicable to symmetric matrices. The duplication matrix $\boldsymbol{D}_p$ is unique and of full column rank. Its central property as applied to the vec operator is $\boldsymbol{D}_p \boldsymbol{L}_p \operatorname{vec}(\boldsymbol{X}) = \operatorname{vec}(\boldsymbol{X})$ for symmetric $\boldsymbol{X}$, and the corresponding Kronecker property is that $\boldsymbol{D}_p \boldsymbol{L}_p (\boldsymbol{X} \otimes \boldsymbol{X}) \boldsymbol{D}_p = (\boldsymbol{X} \otimes \boldsymbol{X}) \boldsymbol{D}_p$ for arbitrary $\boldsymbol{X}$.

Magnus and Neudecker (1979, 1999) also define the *commutation matrix*, which they denote as $\boldsymbol{K}_p$. For any matrix $\boldsymbol{X}$, it is readily apparent that $\operatorname{vec}(\boldsymbol{X})$ and $\operatorname{vec}(\boldsymbol{X}')$ contain the same elements, but in a different order. The commutation matrix is the $p^2 \times p^2$ matrix that performs the transformation $\boldsymbol{K}_p \operatorname{vec}(\boldsymbol{X}) = \operatorname{vec}(\boldsymbol{X}')$ for every $p \times p$ matrix $\boldsymbol{X}$. Henderson and Searle (1979) define an analogous matrix, which they call the *vec-permutation matrix*, which performs the transformation from $\operatorname{vec}(\boldsymbol{X})$ to $\operatorname{vec}(\boldsymbol{X}')$ for any $p_1 \times p_2$ matrix $\boldsymbol{X}$. When $p_1 = p_2$, the definitions are equivalent. The Kronecker property for the commutation matrix $\boldsymbol{K}_p$ is $\boldsymbol{X}_1 \otimes \boldsymbol{X}_2 = \boldsymbol{K}_p (\boldsymbol{X}_2 \otimes \boldsymbol{X}_1) \boldsymbol{K}_p$ for any two $p \times p$ matrices $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$. Informally, the commutation matrix may be used to commute the order of the Kronecker product, hence its name.

Finally, Magnus and Neudecker (1980, 1999) define a fourth matrix, which they simply denote as $\boldsymbol{N}_p$, by $\boldsymbol{N}_p = \frac{1}{2}\left(\boldsymbol{I}_{p^2} + \boldsymbol{K}_p\right)$. Abadir and Magnus (2005, section 11.2) later named this

matrix the *symmetrizer matrix* based on its property applied to the vec operator that $\boldsymbol{N}_p \operatorname{vec}(\boldsymbol{X}) = \frac{1}{2}\operatorname{vec}(\boldsymbol{X} + \boldsymbol{X}')$ for any $p \times p$ matrix $\boldsymbol{X}$. Thus, the symmetrizer matrix transforms an arbitrary square matrix $\boldsymbol{X}$ into a symmetric one, hence its name. The corresponding Kronecker properties for the symmetrizer matrix are

$$\boldsymbol{N}_p(\boldsymbol{X}_1 \otimes \boldsymbol{X}_1)\boldsymbol{N}_p = \boldsymbol{N}_p(\boldsymbol{X}_1 \otimes \boldsymbol{X}_1) = (\boldsymbol{X}_1 \otimes \boldsymbol{X}_1)\boldsymbol{N}_p$$

and

$$\boldsymbol{N}_p(\boldsymbol{X}_1 \otimes \boldsymbol{X}_2 + \boldsymbol{X}_2 \otimes \boldsymbol{X}_1)\boldsymbol{N}_p = \boldsymbol{N}_p(\boldsymbol{X}_1 \otimes \boldsymbol{X}_2 + \boldsymbol{X}_2 \otimes \boldsymbol{X}_1) = (\boldsymbol{X}_1 \otimes \boldsymbol{X}_2 + \boldsymbol{X}_2 \otimes \boldsymbol{X}_1)\boldsymbol{N}_p$$

for any two $p \times p$ matrices $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$. The symmetrizer matrix also has a role in computing certain Kronecker sums, as described below.

The preceding four matrices possess a number of properties useful in the derivation of the multivariate RVM method. The equivalent of Equation (4.7) for the vech operator involves the elimination and duplication matrices and is given by

$$\operatorname{vech}(\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_1') = \boldsymbol{L}_p(\boldsymbol{X}_1 \otimes \boldsymbol{X}_1)\boldsymbol{D}_p \operatorname{vech}\boldsymbol{X}_2 \tag{4.13}$$

for any two matrices $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ such that the product is defined. Other properties given by Magnus and Neudecker (1979, 1980) include:

1. $\boldsymbol{L}_p\boldsymbol{L}_p' = \boldsymbol{I}_{p(p+1)/2}$.
2. $\boldsymbol{L}_p\boldsymbol{D}_p = \boldsymbol{I}_{p(p+1)/2}$.
3. $\boldsymbol{D}_p\boldsymbol{L}_p\boldsymbol{N}_p = \boldsymbol{N}_p$.
4. $\boldsymbol{N}_p = \boldsymbol{N}_p' = \boldsymbol{N}_p^2$; that is, $\boldsymbol{N}_p$ is symmetric and idempotent.
5. For any $p \times p$ matrix $\boldsymbol{X}$, $\boldsymbol{L}_p\boldsymbol{N}_p \operatorname{vec}(\boldsymbol{X}) = \frac{1}{2}\operatorname{vech}(\boldsymbol{X} + \boldsymbol{X}')$. If $\boldsymbol{X}$ is symmetric, then $\operatorname{vech}(\boldsymbol{X}) = \boldsymbol{L}_p\boldsymbol{N}_p \operatorname{vec}(\boldsymbol{X}) = \boldsymbol{L}_p \operatorname{vec}(\boldsymbol{X})$.

6. For any $p \times p$ matrix $\boldsymbol{X}$, $2\boldsymbol{N}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{X}\right)\boldsymbol{D}_p = 2\boldsymbol{N}_p\left(\boldsymbol{X} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p = \left(\boldsymbol{I}_p \otimes \boldsymbol{X} + \boldsymbol{X} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p = \boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{X} + \boldsymbol{X} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p$.

7. For any $p \times p$ matrix $\boldsymbol{X}$, $\boldsymbol{N}_p\left(\boldsymbol{X} \otimes \boldsymbol{X}\right) = \left(\boldsymbol{X} \otimes \boldsymbol{X}\right)\boldsymbol{N}_p = \boldsymbol{N}_p\left(\boldsymbol{X} \otimes \boldsymbol{X}\right)\boldsymbol{N}_p$.

8. For any $p \times p$ matrix $\boldsymbol{X}$, $2\boldsymbol{N}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{X}\right)\boldsymbol{N}_p = 2\boldsymbol{N}_p\left(\boldsymbol{X} \otimes \boldsymbol{I}_p\right)\boldsymbol{N}_p = \boldsymbol{N}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{X} + \boldsymbol{X} \otimes \boldsymbol{I}_p\right) = \left(\boldsymbol{I}_p \otimes \boldsymbol{X} + \boldsymbol{X} \otimes \boldsymbol{I}_p\right)\boldsymbol{N}_p = \boldsymbol{N}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{X} + \boldsymbol{X} \otimes \boldsymbol{I}_p\right)\boldsymbol{N}_p$.

9. For any $p \times p$ nonsingular matrix $\boldsymbol{X}$, $\left|\boldsymbol{L}_p\left(\boldsymbol{X} \otimes \boldsymbol{X}\right)\boldsymbol{D}_p\right|^{-1} = \left|\boldsymbol{L}_p\left(\boldsymbol{X}^{-1} \otimes \boldsymbol{X}^{-1}\right)\boldsymbol{D}_p\right|$.

10. For any $p \times p$ matrix $\boldsymbol{X}$ such that $\boldsymbol{I}_p \otimes \boldsymbol{X} + \boldsymbol{X} \otimes \boldsymbol{I}_p$ is nonsingular, $\left|\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{X} + \boldsymbol{X} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p\right|^{-1} = \left|\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{X} + \boldsymbol{X} \otimes \boldsymbol{I}_p\right)^{-1}\boldsymbol{D}_p\right|$.

11. For any $p \times p$ matrix $\boldsymbol{X}$, $\boldsymbol{D}_p'\operatorname{vec}\left(\boldsymbol{X}\right) = \operatorname{vech}\left(\boldsymbol{X} + \boldsymbol{X}' - \operatorname{diag}\left(\boldsymbol{X}\right)\right)$.

Some additional properties of the $\boldsymbol{L}_p$ and $\boldsymbol{D}_p$ matrices are given in the following lemmas.

Lemma 4.13. *For any $p \times p$ matrix $\boldsymbol{X}_1$ and $p \times p$ symmetric matrix $\boldsymbol{X}_2$,*

$$\left(\boldsymbol{X}_1 \otimes \boldsymbol{X}_1\right)\operatorname{vec}\left(\boldsymbol{X}_2\right) = \boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{X}_1 \otimes \boldsymbol{X}_1\right)\operatorname{vec}\left(\boldsymbol{X}_2\right)$$

and

$$\left(\boldsymbol{X}_1 \otimes \boldsymbol{X}_1\right) = \boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{X}_1 \otimes \boldsymbol{X}_1\right) \tag{4.14}$$

Proof. Applying Equation (4.10) to the symmetric matrix product $\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_1'$ gives

$$\operatorname{vech}\left(\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_1'\right) = \boldsymbol{L}_p\operatorname{vec}\left(\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_1'\right)$$

Applying Equation (4.12) yields

$$\boldsymbol{D}_p\operatorname{vech}\left(\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_1'\right) = \boldsymbol{D}_p\boldsymbol{L}\operatorname{vec}\left(\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_1'\right)$$

or

$$\operatorname{vec}(\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_1') = \boldsymbol{D}_p\boldsymbol{L}\operatorname{vec}(\boldsymbol{X}_1\boldsymbol{X}_2\boldsymbol{X}_1')$$

Using Equation (4.7), this becomes

$$\begin{aligned}(\boldsymbol{X}_1 \otimes \boldsymbol{X}_1)\operatorname{vec}(\boldsymbol{X}_2) &= \boldsymbol{D}_p\boldsymbol{L}\left[(\boldsymbol{X}_1 \otimes \boldsymbol{X}_1)\operatorname{vec}(\boldsymbol{X}_2)\right] \\ &= \boldsymbol{D}_p\boldsymbol{L}(\boldsymbol{X}_1 \otimes \boldsymbol{X}_1)\operatorname{vec}(\boldsymbol{X}_2)\end{aligned}$$

Since no restrictions are placed on $\boldsymbol{X}_2$ apart from symmetry, the column vector $\operatorname{vec}(\boldsymbol{X}_2)$ can be any arbitrary column vector. The second result immediately follows. □

Lemma 4.14. *For any $p \times p$ matrix $\boldsymbol{X}_1$ and $p \times p$ symmetric matrix $\boldsymbol{X}_2$,*

$$\left(\boldsymbol{X}_1 \otimes \boldsymbol{I}_p + \boldsymbol{I}_p \otimes \boldsymbol{X}_1\right)\operatorname{vec}(\boldsymbol{X}_2) = \boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{X}_1 \otimes \boldsymbol{I}_p + \boldsymbol{I}_p \otimes \boldsymbol{X}_1\right)\operatorname{vec}(\boldsymbol{X}_2)$$

and

$$\left(\boldsymbol{X}_1 \otimes \boldsymbol{I}_p + \boldsymbol{I}_p \otimes \boldsymbol{X}_1\right) = \boldsymbol{D}_p\boldsymbol{L}\left(\boldsymbol{X}_1 \otimes \boldsymbol{I}_p + \boldsymbol{I}_p \otimes \boldsymbol{X}_1\right) \tag{4.15}$$

Lemma 4.15. *For any $p \times p$ symmetric matrix $\boldsymbol{X}$ of rank p, $\left|\boldsymbol{L}_p(\boldsymbol{X} \otimes \boldsymbol{X})\boldsymbol{D}_p\right| = |\boldsymbol{X}|^{p+1}$.*

Proof. The proof follows Henderson and Searle (1979) and Magnus and Neudecker (1980). Since X is of full rank, the spectral decomposition $\boldsymbol{X} = \boldsymbol{E}\boldsymbol{L}\boldsymbol{E}'$ exists. Then

$$\begin{aligned}\left|\boldsymbol{L}_p(\boldsymbol{X} \otimes \boldsymbol{X})\boldsymbol{D}_p\right| &= \left|\boldsymbol{L}_p(\boldsymbol{E}\boldsymbol{L}\boldsymbol{E}' \otimes \boldsymbol{E}\boldsymbol{L}\boldsymbol{E}')\boldsymbol{D}_p\right| \\ &= \left|\boldsymbol{L}_p(\boldsymbol{E} \otimes \boldsymbol{E})(\boldsymbol{L} \otimes \boldsymbol{L})(\boldsymbol{E}' \otimes \boldsymbol{E}')\boldsymbol{D}_p\right|\end{aligned}$$

by Property (3) of the Kronecker product. Note that, since $\boldsymbol{L}_p$ and $\boldsymbol{D}_p$ are rectangular matrices,

the determinant cannot be factored into separate determinants. However, using Equation (4.14),

$$\begin{aligned}\left|\boldsymbol{L}_p\left(\boldsymbol{E}\otimes\boldsymbol{E}\right)\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\left(\boldsymbol{E}'\otimes\boldsymbol{E}'\right)\boldsymbol{D}_p\right| &= \left|\boldsymbol{L}_p\left(\boldsymbol{E}\otimes\boldsymbol{E}\right)\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{E}'\otimes\boldsymbol{E}'\right)\boldsymbol{D}_p\right| \\ &= \left|\boldsymbol{L}_p\left(\boldsymbol{E}\otimes\boldsymbol{E}\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{E}'\otimes\boldsymbol{E}'\right)\boldsymbol{D}_p\right|,\end{aligned}$$

which can be factored into three separate determinants as all of the matrices involved are square and of the same size. Then

$$\begin{aligned}&\left|\boldsymbol{L}_p\left(\boldsymbol{E}\otimes\boldsymbol{E}\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{E}'\otimes\boldsymbol{E}'\right)\boldsymbol{D}_p\right| \\ &\qquad= \left|\boldsymbol{L}_p\left(\boldsymbol{E}'\otimes\boldsymbol{E}'\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{E}\otimes\boldsymbol{E}\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right| \\ &\qquad= \left|\boldsymbol{L}_p\left(\boldsymbol{E}'\otimes\boldsymbol{E}'\right)\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{E}\otimes\boldsymbol{E}\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right| \\ &\qquad= \left|\boldsymbol{L}_p\left(\boldsymbol{E}'\otimes\boldsymbol{E}'\right)\left(\boldsymbol{E}\otimes\boldsymbol{E}\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right|\end{aligned}$$

using Equation (4.14) again. Using Property (3) of the Kronecker product,

$$\begin{aligned}\left|\boldsymbol{L}_p\left(\boldsymbol{E}'\otimes\boldsymbol{E}'\right)\left(\boldsymbol{E}\otimes\boldsymbol{E}\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right| &= \left|\boldsymbol{L}_p\left(\boldsymbol{E}'\boldsymbol{E}\otimes\boldsymbol{E}'\boldsymbol{E}\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right| \\ &= \left|\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{I}_p\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right| \\ &= \left|\boldsymbol{L}_p\left(\boldsymbol{I}_{p^2}\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right| \\ &= \left|\boldsymbol{L}_p\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right|. \qquad (4.16)\end{aligned}$$

Using Property (2) of the elimination and duplication matrices,

$$\begin{aligned}\left|\boldsymbol{L}_p\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right| &= \left|\boldsymbol{I}_{p(p+1)/2}\right|\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right| \\ &= \left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right| \qquad (4.17)\end{aligned}$$

Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ be the eigenvalues of $\boldsymbol{L}$, which are the same as its diagonal elements, and let $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_p$ be the corresponding eigenvectors. Henderson and Searle (1979) simply note that $\boldsymbol{L}\otimes\boldsymbol{L}$ is a diagonal matrix with elements $\lambda_i\lambda_j$, $i, j = 1, 2, \ldots, p$, along the diagonal. Then

$\boldsymbol{L}_p(\boldsymbol{L}\otimes\boldsymbol{L})\boldsymbol{D}_p$ is equal to $\boldsymbol{L}_p$ times $\boldsymbol{D}_p$ with the $[p(i-1)+j]$th row of $\boldsymbol{D}_p$ multiplied by $\lambda_i\lambda_j$, which gives the formula for the determinant. However, the determinant may be evaluated by finding the eigenvalues of $\boldsymbol{L}_p(\boldsymbol{L}\otimes\boldsymbol{L})\boldsymbol{D}_p$. Note that the $\frac{1}{2}p(p+1)$ eigenvalues of $\boldsymbol{L}_p(\boldsymbol{L}\otimes\boldsymbol{L})\boldsymbol{D}_p$ are the same as the $\frac{1}{2}p(p+1)$ nonzero eigenvalues of $\boldsymbol{D}_p\boldsymbol{L}_p(\boldsymbol{L}\otimes\boldsymbol{L})$. Then

$$\begin{aligned}\boldsymbol{D}_p\boldsymbol{L}_p(\boldsymbol{L}\otimes\boldsymbol{L})\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j+\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)&=\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{L}\boldsymbol{e}_i\otimes\boldsymbol{L}\boldsymbol{e}_j+\boldsymbol{L}\boldsymbol{e}_j\otimes\boldsymbol{L}\boldsymbol{e}_i\right)\\&=\boldsymbol{D}_p\boldsymbol{L}_p\left(\lambda_i\boldsymbol{e}_i\otimes\lambda_j\boldsymbol{e}_j+\lambda_j\boldsymbol{e}_j\otimes\lambda_i\boldsymbol{e}_i\right)\end{aligned}$$

using the definition of eigenvalues. Using Property (1) of Kronecker products,

$$\begin{aligned}\boldsymbol{D}_p\boldsymbol{L}_p\left(\lambda_i\boldsymbol{e}_i\otimes\lambda_j\boldsymbol{e}_j+\lambda_j\boldsymbol{e}_j\otimes\lambda_i\boldsymbol{e}_i\right)&=\boldsymbol{D}_p\boldsymbol{L}_p\left(\lambda_i\lambda_j\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j\right)+\lambda_j\lambda_i\left(\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)\right)\\&=\lambda_i\lambda_j\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j+\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)\end{aligned}$$

Applying Equation (4.8) yields

$$\lambda_i\lambda_j\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j+\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)=\lambda_i\lambda_j\boldsymbol{D}_p\boldsymbol{L}_p\,\mathrm{vec}\left(\boldsymbol{e}_j\boldsymbol{e}_i'+\boldsymbol{e}_i\boldsymbol{e}_j'\right)$$

The results of Equation (4.14) then imply

$$\lambda_i\lambda_j\boldsymbol{D}_p\boldsymbol{L}_p\,\mathrm{vec}\left(\boldsymbol{e}_j\boldsymbol{e}_i'+\boldsymbol{e}_i\boldsymbol{e}_j'\right)=\lambda_i\lambda_j\,\mathrm{vec}\left(\boldsymbol{e}_j\boldsymbol{e}_i'+\boldsymbol{e}_i\boldsymbol{e}_j'\right)$$

and a second application of Equation (4.8) gives

$$\lambda_i\lambda_j\,\mathrm{vec}\left(\boldsymbol{e}_j\boldsymbol{e}_i'+\boldsymbol{e}_i\boldsymbol{e}_j'\right)=\lambda_i\lambda_j\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j+\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right).$$

Then by definition $\lambda_i\lambda_j, i=1,2,\ldots,p, j=i,i+1,\ldots,p$, are the $\frac{1}{2}p(p+1)$ nonzero eigenvalues of $\boldsymbol{D}_p\boldsymbol{L}_p(\boldsymbol{L}\otimes\boldsymbol{L})$, which are also the eigenvalues of $\boldsymbol{L}_p(\boldsymbol{L}\otimes\boldsymbol{L})\boldsymbol{D}_p$. Using Property (2) of

eigenvalues,

$$\left|\boldsymbol{L}_p\left(\boldsymbol{L}\otimes\boldsymbol{L}\right)\boldsymbol{D}_p\right| = \prod_{i\le j}^{p}\lambda_i\lambda_j$$

$$= \prod_{i=1}^{p}\lambda_i^{p+1}$$

$$= \left(\prod_{i=1}^{p}\lambda_i\right)^{p+1}$$

$$= |\boldsymbol{X}|^{p+1}$$

□

Lemma 4.16. *For any $p \times p$ nonsingular symmetric matrix $\boldsymbol{X}$, $\left|\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{X}+\boldsymbol{X}\otimes\boldsymbol{I}_p\right)\boldsymbol{D}_p\right| = \prod_{i\le j}^{p}\left(\lambda_i+\lambda_j\right) = 2^p\,|\boldsymbol{X}|\prod_{i<j}^{p}\left(\lambda_i+\lambda_j\right)$, where $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_p$ are the eigenvalues of $\boldsymbol{X}$.*

Proof. The proof is sketched in Magnus and Neudecker (1980) and is similar to Lemma 4.15. The determinant is evaluated by finding the $\frac{1}{2}p\,(p+1)$ eigenvalues of $\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{X}+\boldsymbol{X}\otimes\boldsymbol{I}_p\right)\boldsymbol{D}_p$, which are the same as the $\frac{1}{2}p\,(p+1)$ nonzero eigenvalues of $\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{X}+\boldsymbol{X}\otimes\boldsymbol{I}_p\right)$. Let $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_p$ be the eigenvalues of $\boldsymbol{X}$ and let $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_p$ be the corresponding eigenvectors. Then

$$\begin{aligned}
&\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{X}+\boldsymbol{X}\otimes\boldsymbol{I}_p\right)\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j+\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)\\
&\quad= \boldsymbol{D}_p\boldsymbol{L}_p\left[\left(I_p\otimes X\right)\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j\right)+\left(\boldsymbol{I}_p\otimes\boldsymbol{X}\right)\left(\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)+\left(\boldsymbol{X}\otimes\boldsymbol{I}_p\right)\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j\right)+\left(\boldsymbol{X}\otimes\boldsymbol{I}_p\right)\left(\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)\right]\\
&\quad= \boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{I}_p\boldsymbol{e}_i\otimes\boldsymbol{X}\boldsymbol{e}_j+\boldsymbol{I}_p\boldsymbol{e}_j\otimes\boldsymbol{X}\boldsymbol{e}_i+\boldsymbol{X}\boldsymbol{e}_i\otimes\boldsymbol{I}_p\boldsymbol{e}_j+\boldsymbol{X}\boldsymbol{e}_j\otimes\boldsymbol{I}_p\boldsymbol{e}_i\right)\\
&\quad= \boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{e}_i\otimes\lambda_j\boldsymbol{e}_j+\boldsymbol{e}_j\otimes\lambda_i\boldsymbol{e}_i+\lambda_i\boldsymbol{e}_i\otimes\boldsymbol{e}_j+\lambda_j\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)
\end{aligned}$$

using the definition of eigenvalues and Properties (2) and (3) of Kronecker products. Using Property (1) of Kronecker products,

$$\begin{aligned}
&\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{e}_i\otimes\lambda_j\boldsymbol{e}_j+\boldsymbol{e}_j\otimes\lambda_i\boldsymbol{e}_i+\lambda_i\boldsymbol{e}_i\otimes\boldsymbol{e}_j+\lambda_j\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)\\
&\quad=\boldsymbol{D}_p\boldsymbol{L}_p\left[\lambda_j\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j\right)+\lambda_i\left(\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)+\lambda_i\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j\right)+\lambda_j\left(\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)\right]\\
&\quad=\boldsymbol{D}_p\boldsymbol{L}_p\left[\left(\lambda_i+\lambda_j\right)\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j\right)+(\lambda_i+\lambda_i)\left(\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)\right]\\
&\quad=\left(\lambda_i+\lambda_j\right)\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j+\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)
\end{aligned}$$

Applying Equation (4.8) yields

$$\left(\lambda_i+\lambda_j\right)\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j+\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right)=\left(\lambda_i+\lambda_j\right)\boldsymbol{D}_p\boldsymbol{L}_p\,\mathrm{vec}\left(\boldsymbol{e}_j\boldsymbol{e}_i'+\boldsymbol{e}_i\boldsymbol{e}_j'\right)$$

The results of Equation (4.14) then imply

$$\left(\lambda_i+\lambda_j\right)\boldsymbol{D}_p\boldsymbol{L}_p\,\mathrm{vec}\left(\boldsymbol{e}_j\boldsymbol{e}_i'+\boldsymbol{e}_i\boldsymbol{e}_j'\right)=\left(\lambda_i+\lambda_j\right)\mathrm{vec}\left(\boldsymbol{e}_j\boldsymbol{e}_i'+\boldsymbol{e}_i\boldsymbol{e}_j'\right)$$

and a second application of Equation (4.8) gives

$$\left(\lambda_i+\lambda_j\right)\mathrm{vec}\left(\boldsymbol{e}_j\boldsymbol{e}_i'+\boldsymbol{e}_i\boldsymbol{e}_j'\right)=\left(\lambda_i+\lambda_j\right)\left(\boldsymbol{e}_i\otimes\boldsymbol{e}_j+\boldsymbol{e}_j\otimes\boldsymbol{e}_i\right).$$

Then by definition $\lambda_i+\lambda_j, i=1,2,\ldots,p, j=i,i+1,\ldots,p$, are the nonzero eigenvalues of $\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{X}+\boldsymbol{X}\otimes\boldsymbol{I}_p\right)$, which are also the eigenvalues of $\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{X}+\boldsymbol{X}\otimes\boldsymbol{I}_p\right)\boldsymbol{D}_p$. Using Prop-

erty (2) of eigenvalues,

$$\begin{aligned}\left|\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{X} + \boldsymbol{X} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p\right| &= \prod_{i\leq j}^{p}\left(\lambda_i + \lambda_j\right) \\ &= \prod_{i=1}^{p}(\lambda_i + \lambda_i) \cdot \prod_{i<j}^{p}\left(\lambda_i + \lambda_j\right) \\ &= \prod_{i=1}^{p} 2\lambda_i \cdot \prod_{i<j}^{p}\left(\lambda_i + \lambda_j\right) \\ &= 2^p \prod_{i=1}^{p}\lambda_i \cdot \prod_{i<j}^{p}\left(\lambda_i + \lambda_j\right) \\ &= 2^p\,|\boldsymbol{X}| \prod_{i<j}^{p}\left(\lambda_i + \lambda_j\right)\end{aligned}$$

□

4.3.1.6 Jacobian Determinant

Another useful concept in the study of multivariate statistics is that of the *Jacobian determinant* which will be used extensively in the development of the multivariate RVM method.

Definition 4.17. ***Jacobian Determinant.*** *Let the matrix $\boldsymbol{Y} = \boldsymbol{F}(\boldsymbol{X})$ be a $p_1 \times p_2$ matrix of differentiable functions of the elements of the $p_3 \times p_4$ random matrix $\boldsymbol{X}$. The function $\boldsymbol{F}(\cdot)$ is called a transformation from $\boldsymbol{X}$ to $\boldsymbol{Y}$, denoted $\boldsymbol{X} \to \boldsymbol{Y}$. If the transformation $\boldsymbol{X} \to \boldsymbol{Y}$ is one-to-one, then the Jacobian determinant (or simply Jacobian) will be denoted $J_{\boldsymbol{X}\to\boldsymbol{Y}}$ and is defined as the absolute value of the determinant*

$$\left|\frac{\partial x_{ij}}{\partial y_{kl}}\right|,$$

where $\boldsymbol{X} = \{x_{ij}\}$, $i = 1, 2, \ldots, p_3$, $j = 1, 2, \ldots, p_4$; and $\boldsymbol{Y} = \{y_{kl}\}$, $k = 1, 2, \ldots, p_1$, $l = 1, 2, \ldots, p_2$, are the elements of $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively.

In statistics, Jacobians arise naturally in obtaining the distribution for a transformation of random variables. Again consider the transformation $\boldsymbol{X} \to \boldsymbol{Y}$, which is given by the function

$\boldsymbol{Y} = \boldsymbol{F}(\boldsymbol{X})$. Since the transformation is one-to-one, there also exists an inverse function $\boldsymbol{F}^{-1}$ such that $\boldsymbol{X} = \boldsymbol{F}^{-1}(\boldsymbol{Y})$. Let the probability density function of $\boldsymbol{X}$ be given by the scalar function $f_{\boldsymbol{X}}(\boldsymbol{X})$; then the probability density of $f_{\boldsymbol{Y}}(\boldsymbol{Y})$ is given by

$$f_{\boldsymbol{Y}}(\boldsymbol{Y}) = f_{\boldsymbol{Y}}(\boldsymbol{F}(\boldsymbol{X})) = f_{\boldsymbol{X}}\left(\boldsymbol{F}^{-1}(\boldsymbol{Y})\right) J_{\boldsymbol{X}\to\boldsymbol{Y}}$$

As defined above, the computation of the Jacobian determinant involves the computation of $p_1 p_2 p_3 p_4$ partial derivatives. This potentially daunting task may be simplified by the use of matrix derivatives and matrix differentials (Magnus and Neudecker, 1999, chapter 9), which are discussed next.

4.3.1.7 Matrix Derivative and Differential

For scalar functions of scalar variables, the (first) derivative of a function $y = y(x)$ with respect to the variable x, denoted $\frac{dy}{dx}$, is well defined as

$$\frac{dy}{dx} = \lim_{h\to 0} \frac{y(x+h) - y(x)}{h}$$

at the point x. An equivalent expression to Equation (4.18) is

$$y(x+h) = y(x) + h\frac{dy}{dx} + r(x) \tag{4.18}$$

where $r(\cdot)$ is the *remainder function* satisfying the equation

$$\lim_{h\to 0} \frac{r(h)}{h} = 0.$$

The (first) differential of the scalar function y, denoted as dy, is then defined as

$$dy(x;h) = h\frac{dy}{dx} \tag{4.19}$$

Equations (4.18) and (4.19) are easily generalized to the definition of a matrix derivative. The partial derivatives $\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \ldots, \frac{\partial y}{\partial x_p}$ are also well defined when $y = y\left(x_1, x_2, \ldots, x_p\right)$ is a function of p variables. Similarly, if y is a scalar function of a vector $\boldsymbol{x} = [x_1 x_2 \ldots x_p]'$, then the definition of $\frac{\partial y}{\partial \boldsymbol{x}'}$, the derivative of y with respect to $\boldsymbol{x}$, is the $1 \times p$ vector

$$\frac{\partial y}{\partial \boldsymbol{x}'} = \begin{bmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} & \cdots & \frac{\partial y}{\partial x_p} \end{bmatrix}.$$

This definition is generally accepted, although some sources, such as Rencher (2000, p. 51), define the derivative as a column rather than a row vector. If y is a scalar function of the $p_2 \times p_1$ matrix $\boldsymbol{X} = \left(x_{ij}\right)$, then

$$\frac{\partial y}{\partial \boldsymbol{x}'} = \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \frac{\partial y}{\partial x_{12}} & \cdots & \frac{\partial y}{\partial x_{1,p_1}} \\ \frac{\partial y}{\partial x_{21}} & \frac{\partial y}{\partial x_{22}} & \cdots & \frac{\partial y}{\partial x_{2,p_1}} \\ \frac{\partial y}{\partial x_{31}} & \frac{\partial y}{\partial x_{32}} & \cdots & \frac{\partial y}{\partial x_{3,p_1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial x_{p_2,1}} & \frac{\partial y}{\partial x_{p_2,2}} & \cdots & \frac{\partial y}{\partial x_{p_2,p_1}} \end{bmatrix} \tag{4.20}$$

If $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x})$ is a $p_2 \times 1$ vector-valued function of the $p_1 \times 1$ vector $\boldsymbol{x}$, then let the individual elements of $\boldsymbol{x}$ be denoted as $\boldsymbol{x} = \{x_i\}$, $i = 1, 2, \ldots, p_1$, and the individual elements of $\boldsymbol{y}$ be denoted $\boldsymbol{y} = \{y_j\}$, $j = 1, 2, \ldots, p_2$. The derivative (or Jacobian matrix) of $\boldsymbol{y}$ with respect to $\boldsymbol{x}$ is defined as

the $p_2 \times p_1$ matrix

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}'} = \left[\frac{\partial y_j}{\partial x_i}\right]$$
$$= \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_{p_1}} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_{p_1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_{p_2}}{\partial x_1} & \frac{\partial y_{p_2}}{\partial x_2} & \cdots & \frac{\partial y_{p_2}}{\partial x_{p_1}} \end{bmatrix}$$

For a matrix derivative, let $\boldsymbol{X}$ be a $p_1 \times p_2$ matrix, and let $\boldsymbol{Y} = \boldsymbol{F}(\boldsymbol{X})$ be a $p_3 \times p_4$ matrix of differentiable functions of the elements of $\boldsymbol{X}$. These individual elements will be denoted $\boldsymbol{X} = \{x_{ij}\}$, $i = 1, 2, \ldots, p_1$, $j = 1, 2, \ldots, p_2$, for $\boldsymbol{X}$ and $\boldsymbol{Y} = \{y_{kl}\}$, $k = 1, 2, \ldots, p_3$, $l = 1, 2, \ldots, p_4$, for $\boldsymbol{Y}$. Then the matrix derivative $\dfrac{\partial \boldsymbol{Y}}{\partial \boldsymbol{X}}$ denotes the $p_1 p_2 p_3 p_4$ partial derivatives $\dfrac{\partial y_{kl}}{\partial x_{ij}}$. However, although there is agreement that the matrix derivative should include all of these partial derivatives, there are several different notations that have been used that correspond to a different ordering of these partial derivatives. Magnus and Neudecker (1985) argue strongly for the following definition, based on several considerations, including the existence of a meaningful chain rule.

Definition 4.18. ***Matrix Derivative.*** *Let $\boldsymbol{X}$ be a $p_1 \times p_2$ matrix, and let $\boldsymbol{Y} = \boldsymbol{F}(\boldsymbol{X})$ be a $p_3 \times p_4$ matrix of differentiable functions of the elements of $\boldsymbol{X}$. The matrix derivative (or Jacobian matrix) of $\boldsymbol{Y}$ at $\boldsymbol{X}$ is the $p_1 p_2 \times p_3 p_4$ matrix $d\boldsymbol{F}(\boldsymbol{X})$ given by*

$$d\boldsymbol{F}(\boldsymbol{X}) = \frac{\partial \operatorname{vec}(\boldsymbol{F}(\boldsymbol{X}))}{\partial (\operatorname{vec}(\boldsymbol{X}))'} \tag{4.21}$$

This reduces the computation of a matrix derivative to the computation of a vector derivative, for which a well-established definition exists. Furthermore, the Jacobian determinant may be calculated as the determinant of the Jacobian matrix. For symmetric matrices, a similar notation

exists:

$$d\boldsymbol{F}(\boldsymbol{X}) = \frac{\partial \operatorname{vech}(\boldsymbol{F}(\boldsymbol{X}))}{\partial(\operatorname{vech}(\boldsymbol{X}))}$$

Magnus and Neudecker (1980, 1999) originally used the operator $\upsilon(\cdot)$ instead of $\operatorname{vech}(\cdot)$, but later works adopted the $\operatorname{vech}(\cdot)$ notation (Abadir and Magnus, 2005). The latter will be used throughout this work as it seems to be in more common usage and also avoids any ambiguity with the variable v, but otherwise the notation of Magnus and Neudecker (1999) will be used.

Magnus and Neudecker (1999) also describe the concept of the *matrix differential*, which is similar to the differential of a scalar function of scalar variables in Equation (4.19). Let $\boldsymbol{F}\colon E \to \mathbb{R}^{p_3 \times p_4}$ be a matrix function defined on the set $E \subseteq \mathbb{R}^{p_1 \times p_2}$, and let the matrix $\boldsymbol{X}$ be an interior point of E. The scalar distance measure in Equation (4.18) is replaced by the usual matrix norm, which is denoted by $\| \cdot \|$ and defined as $\| \boldsymbol{X} \| = \sqrt{\operatorname{Tr}(\boldsymbol{X}'\boldsymbol{X})}$ for the matrix $\boldsymbol{X}$. The interval $y(x+h) - y(x)$ in Equation (4.18) is replaced by $B(\boldsymbol{X}, r) \subset E$, where $B(\cdot,\cdot)$ is the ball (or neighborhood) of E with center $\boldsymbol{X}$ and radius rd, consisting of those points in E where $B(\boldsymbol{X}, rd) = (\boldsymbol{X}\colon \boldsymbol{X} \in \mathbb{R}^{p_1 \times p_2}, \| \boldsymbol{X} - \boldsymbol{C} \| < rd)$. Let $\boldsymbol{U}$ be a point in $\mathbb{R}^{p_1 \times p_2}$ with $\| \boldsymbol{U} \| < rd$, so that $\boldsymbol{X} + \boldsymbol{U} \in B(\boldsymbol{X}, rd)$. Then the matrix differential $d\boldsymbol{F}(\boldsymbol{X}; \boldsymbol{U})$ is a number which approximates the value of the function $\boldsymbol{F}(\boldsymbol{X} + \boldsymbol{U})$ by the affine function $\boldsymbol{F}(\boldsymbol{X}) + d\boldsymbol{F}(\boldsymbol{X}; \boldsymbol{U})\boldsymbol{U}$. In other words, the function of the sum $\boldsymbol{X} + \boldsymbol{U}$ can be approximated by the value of the function at $\boldsymbol{X}$ plus another value that is proportional to $\boldsymbol{U}$. The formal definition of the matrix differential is as follows.

Definition 4.19. ***Matrix Differential.*** *If there exists a real $p_1 p_3 \times p_2 p_4$ matrix $\boldsymbol{A}$, depending on $\boldsymbol{X}$ but not on $\boldsymbol{U}$, such that*

$$\operatorname{vec}(\boldsymbol{F}(\boldsymbol{X} + \boldsymbol{U})) = \operatorname{vec}(\boldsymbol{F}(\boldsymbol{X})) + \boldsymbol{A}(\boldsymbol{X}) \operatorname{vec}(\boldsymbol{U}) + \operatorname{vec}(\boldsymbol{R}_c(\boldsymbol{U}))$$

for all $\boldsymbol{U} \in \mathbb{R}^{p_1 \times p_2}$ with $\| \boldsymbol{U} \| < r$ and the remainder function $\boldsymbol{R}_c$ satisfying

$$\lim_{\boldsymbol{U} \to \boldsymbol{0}} \frac{\boldsymbol{R}_c(\boldsymbol{U})}{\| \boldsymbol{U} \|} = \boldsymbol{0},$$

then the function $\boldsymbol{F}$ *is said to be differentiable at* $\boldsymbol{X}$*. Furthermore, the* $p_3 \times p_4$ *matrix* $d\boldsymbol{F}(\boldsymbol{X}; \boldsymbol{U})$ *given by*

$$\operatorname{vec}(d\boldsymbol{F}(\boldsymbol{X}; \boldsymbol{U})) = \boldsymbol{A}(\boldsymbol{X}) \operatorname{vec}(\boldsymbol{U})$$

is called the (first) differential of $\boldsymbol{F}$ *at* $\boldsymbol{X}$ *with increment* $\boldsymbol{U}$ *and the* $p_1 p_3 \times p_2 p_4$ *matrix* $\boldsymbol{A}(\boldsymbol{X})$ *is called the (first) derivative of* $\boldsymbol{F}$ *at* $\boldsymbol{X}$*.*

Magnus and Neudecker (1999, pp. 148 and 172) also provide a list of several properties for the matrix differential; most of these follow the equivalent expressions for scalar functions of scalar variables:

1. $d\boldsymbol{C} = \boldsymbol{0}$
2. $d\boldsymbol{X} = \boldsymbol{I}_{p_1 p_2}$
3. $d(\boldsymbol{G} + \boldsymbol{H}) = d\boldsymbol{G} + d\boldsymbol{H}$
4. $d(\boldsymbol{G}\boldsymbol{H}) = (d\boldsymbol{G})\boldsymbol{H} + \boldsymbol{G}(d\boldsymbol{H})$ (Product Rule)
5. $d(\boldsymbol{G}(\boldsymbol{H})) = d\boldsymbol{G}(\boldsymbol{H})\, d\boldsymbol{H}$ (Chain Rule)
6. $d(\boldsymbol{G} \otimes \boldsymbol{H}) = d\boldsymbol{G} \otimes d\boldsymbol{H}$
7. $d\boldsymbol{G}' = (d\boldsymbol{G})'$
8. $d \operatorname{vec}(\boldsymbol{G}) = \operatorname{vec}(d\boldsymbol{G})$
9. $d \operatorname{Tr}(\boldsymbol{G}) = \operatorname{Tr}(d\boldsymbol{G})$

where $\boldsymbol{G}(\cdot)$ and $\boldsymbol{H}(\cdot)$ are matrix functions of the $p_1 \times p_2$ matrix $\boldsymbol{X}$, and $\boldsymbol{C}$ is a matrix constant.

The value of matrix differentials in the present work is in the computation of Jacobian matrices and Jacobian determinants. This is due to the following theorem by Magnus and Neudecker (1999).

Theorem 4.20 (First Identification Theorem)**.** *Let* $\boldsymbol{F} : S \to \mathbb{R}^{p_3 \times p_4}$ *be a matrix function on a set* $E \subseteq R^{p_1 \times p_2}$*, and differentiable at an interior point* $\boldsymbol{X} \in E$*. Then*

$$\operatorname{vec}\left(d\boldsymbol{F}\left(\boldsymbol{X};\boldsymbol{U}\right)\right) = \boldsymbol{A}\left(\boldsymbol{X}\right)\operatorname{vec}\left(\boldsymbol{U}\right)$$

for all $\boldsymbol{U} \in \mathbb{R}^{p_3 \times p_4}$ *if and only if*

$$J_{\boldsymbol{F}\to\boldsymbol{X}}\left(\boldsymbol{X}\right) = \boldsymbol{A}\left(\boldsymbol{X}\right),$$

or, equivalently,

$$J_{\boldsymbol{X}\to\boldsymbol{F}}\left(\boldsymbol{X}\right) = \boldsymbol{A}^{-1}\left(\boldsymbol{X}\right).$$

The First Identification Theorem relates the Jacobian matrix, from which the Jacobian determinant is calculated, to the matrix differential. This greatly simplifies the computation of the Jacobian determinant compared to the computation using the partial derivatives of the individual elements. Given a matrix function $\boldsymbol{Y} = \boldsymbol{F}\left(\boldsymbol{X}\right)$, one must compute the differential $d\boldsymbol{F}\left(\boldsymbol{X}\right)$, then apply the vec (or vech) operator to obtain an equation of the form $d\left(\operatorname{vec}\left(\boldsymbol{F}\left(\boldsymbol{X}\right)\right)\right) = \boldsymbol{A}\left(\boldsymbol{X}\right)d\left(\operatorname{vec}\left(\boldsymbol{X}\right)\right)$. It can then be concluded that $J_{\boldsymbol{X}\to\boldsymbol{Y}} = \boldsymbol{A}^{-1}\left(\boldsymbol{X}\right)$.

The set of lemmas in the next subsection illustrate the application of the First Identification Theorem by the computation of several Jacobian determinants that will be needed in the derivation of the multivariate RVM method. The following properties of the Jacobian determinant and the matrix differential, outlined by Olkin and Sampson (1972), will be used in these and other proofs:

1. If $\boldsymbol{Y}_1 = \boldsymbol{F}\left(\boldsymbol{X}\right)$ and $\boldsymbol{Y}_2 = \boldsymbol{G}\left(\boldsymbol{Y}_1\right) = \boldsymbol{G}\left(\boldsymbol{F}\left(\boldsymbol{X}\right)\right)$ are both matrix functions, then $J_{\boldsymbol{X}\to\boldsymbol{Y}_2} = J_{\boldsymbol{X}\to\boldsymbol{Y}_1}J_{\boldsymbol{Y}_1\to\boldsymbol{Y}_2}$.

2. If $\boldsymbol{Y} = \boldsymbol{F}\left(\boldsymbol{X}\right)$ is a matrix function and $J_{\boldsymbol{X}\to\boldsymbol{Y}} \neq 0$, then $J_{\boldsymbol{X}\to\boldsymbol{Y}}J_{\boldsymbol{Y}\to\boldsymbol{X}} = 1$. Alternatively, $J_{\boldsymbol{X}\to\boldsymbol{Y}} = 1/J_{\boldsymbol{Y}\to\boldsymbol{X}}$.

3. If $\boldsymbol{Y}_1 = \boldsymbol{F}\left(\boldsymbol{X}_1\right)$ and $\boldsymbol{Y}_2 = \boldsymbol{G}\left(\boldsymbol{X}_2\right)$ are matrix functions whose arguments $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$ do not

depend on each other, then $J_{\boldsymbol{X}_1,\boldsymbol{X}_2\to\boldsymbol{Y}_1,\boldsymbol{Y}_2} = J_{\boldsymbol{X}_1\to\boldsymbol{Y}_1} J_{\boldsymbol{X}_2\to\boldsymbol{Y}_2}$.

4. If $\boldsymbol{Y} = \boldsymbol{F}(\boldsymbol{X})$ is a smooth transformation of $\boldsymbol{X}$ to $\boldsymbol{Y}$, then $J_{\boldsymbol{X}\to\boldsymbol{Y}} = J_{d\boldsymbol{X}\to d\boldsymbol{Y}}$.

Property (1) allows a complex Jacobian to be broken into a series of simpler steps that are easier to compute. Property (2) allows a Jacobian to be found from the Jacobian of the inverse transformation, which may be easier to compute. Finally, Property (4) means that the Jacobian of a nonlinear transformation may still be computed if the transformation is linear in the differentials.

4.3.2 Results for Derivatives and Jacobian Matrices

A number of specific matrix derivatives and Jacobian matrices are needed in the development of the multivariate RVM. These are stated in the following lemmas.

4.3.2.1 Scalar Functions of a Matrix

Lemma 4.21. *Let $\boldsymbol{X}$ be a $p \times p$ matrix, and let y be the scalar function defined by $y = |\boldsymbol{X}|$. Then, at the points where $\boldsymbol{X}$ is nonsingular,*

$$dy = |\boldsymbol{X}| \operatorname{Tr}\left(\boldsymbol{X}^{-1} d\boldsymbol{X}\right) \tag{4.22}$$

and

$$\frac{\partial y}{\partial [\operatorname{vec}(\boldsymbol{X})]'} = |\boldsymbol{X}| \left[\operatorname{vec}\left(\boldsymbol{X}^{-1\prime}\right)\right]'.$$

Proof. The proof is outlined in Magnus and Neudecker (1999). In this case, the element-by-element definition of the matrix derivative in Equation (4.20), rather than the definition of the derivative based on matrices in Equation (4.21), must be applied. From the alternative definition of the determinant in terms of cofactors in Equation (4.5),

$$|\boldsymbol{X}| = \sum_{j=1}^{p} cf_{ij} x_{ij} \quad (i = 1, 2, \ldots, p).$$

From the definition of the cofactor, it is apparent that cf_{ij} does not depend on x_{ij} for any given j. Thus

$$\begin{aligned}\frac{\partial y}{\partial x_{ij}} &= \frac{\partial}{\partial x_{ij}} \sum_{j=1}^{p} cf_{ij} x_{ij} \\ &= cf_{ij}\end{aligned}$$

for all i, so that

$$\begin{aligned}dy &= \sum_{i=1}^{p} \sum_{j=1}^{p} cf_{ij} dx_{ij} \\ &= \operatorname{Tr}\left(\boldsymbol{X}^{\#} d\boldsymbol{X}\right)\end{aligned} \tag{4.23}$$

From Property (4) of determinants, $\boldsymbol{X}^{\#}\boldsymbol{X} = |\boldsymbol{X}|\,\boldsymbol{I}_p \Rightarrow \boldsymbol{X}^{\#} = |\boldsymbol{X}|\,\boldsymbol{X}^{-1}$. Substituting into Equation (4.23) and applying Property (3) of the trace gives

$$dy = |\boldsymbol{X}| \operatorname{Tr}\left(\boldsymbol{X}^{-1} d\boldsymbol{X}\right).$$

Using the results of Equation (4.9), this becomes

$$dy = |\boldsymbol{X}| \left[\operatorname{vec}\left(\boldsymbol{X}^{-1\prime}\right)\right]' \operatorname{vec}\left(d\boldsymbol{X}\right).$$

By the First Identification Theorem, $d\,|\boldsymbol{X}| = |\boldsymbol{X}| \operatorname{Tr}\left(\boldsymbol{X}^{-1}\right) d\boldsymbol{X}$ and $\dfrac{\partial\,|\boldsymbol{X}|}{\partial\,[\operatorname{vec}(\boldsymbol{X})]'} = |\boldsymbol{X}| \left[\operatorname{vec}\left(\boldsymbol{X}^{-1\prime}\right)\right]'$.

□

Lemma 4.22. *Let $\boldsymbol{X}$ be a $p \times p$ matrix, and let y be the scalar function defined by $y = \log(|\boldsymbol{X}|)$. Then, at the points where $|\boldsymbol{X}| > 0$,*

$$dy = \operatorname{Tr}\left(\boldsymbol{X}^{-1} d\boldsymbol{X}\right) \tag{4.24}$$

and

$$\frac{\partial y}{\partial \left[\text{vec}\left(\boldsymbol{X}\right)\right]'} = \left[\text{vec}\left(\boldsymbol{X}^{-1\prime}\right)\right]'.$$

Proof. The proof is outlined in Abadir and Magnus (2005, Section 13.2). Using the Chain Rule (Property (5) of the matrix differential) and the results of Lemma 4.21,

$$\begin{aligned} dy &= \frac{1}{|\boldsymbol{X}|} d\left|\boldsymbol{X}\right| \\ &= \frac{1}{|\boldsymbol{X}|} \left|\boldsymbol{X}\right| \text{Tr}\left(\boldsymbol{X}^{-1} d\boldsymbol{X}\right) \\ &= \text{Tr}\left(\boldsymbol{X}^{-1} d\boldsymbol{X}\right) \\ &= \left[\text{vec}\left(\boldsymbol{X}^{-1\prime}\right)\right]' \text{vec}\left(d\boldsymbol{X}\right) \end{aligned}$$

By the First Identification Theorem, $d\log\left(|\boldsymbol{X}|\right) = \text{Tr}\left(\boldsymbol{X}^{-1}\right) d\boldsymbol{X}$ and $\frac{\partial \log\left(|\boldsymbol{X}|\right)'}{\partial \left[\text{vec}\left(\boldsymbol{X}\right)\right]'} = \left[\text{vec}\left(\boldsymbol{X}^{-1\prime}\right)\right]'$. □

Lemma 4.23. *Let $\boldsymbol{X}$ be a $p \times p$ matrix and let $\boldsymbol{C}$ be $p \times p$ matrix. Let y be the scalar function defined by $y = \text{Tr}\left(\boldsymbol{C}\boldsymbol{X}'\boldsymbol{X}\right)$. Then*

$$dy = \text{Tr}\left[\left(\boldsymbol{C} + \boldsymbol{C}'\right)\boldsymbol{X}' d\boldsymbol{X}\right]$$

and

$$\frac{\partial y}{\partial \left[\text{vec}\left(\boldsymbol{X}\right)\right]'} = \left(\text{vec}\left[\boldsymbol{X}\left(\boldsymbol{C} + \boldsymbol{C}'\right)\right]\right)'.$$

Proof. The proof is similar to that of Abadir and Magnus (2005, Section 13.2). Using Proper-

ties (4), (7), and (9) of the matrix differential,

$$\begin{aligned} dy &= d\,\mathrm{Tr}\left(\boldsymbol{C}\boldsymbol{X}'\boldsymbol{X}\right) \\ &= \mathrm{Tr}\left[d\left(\boldsymbol{C}\boldsymbol{X}'\boldsymbol{X}\right)\right] \\ &= \mathrm{Tr}\left[\boldsymbol{C}\left(d\boldsymbol{X}'\right)\boldsymbol{X} + \boldsymbol{C}\boldsymbol{X}'d\boldsymbol{X}\right] \\ &= \mathrm{Tr}\left[\boldsymbol{C}\left(d\boldsymbol{X}\right)'\boldsymbol{X} + \boldsymbol{C}\boldsymbol{X}'d\boldsymbol{X}\right]. \end{aligned}$$

Using Properties (1), (2), and (4) of the trace,

$$\begin{aligned} \mathrm{Tr}\left[\boldsymbol{C}\left(d\boldsymbol{X}\right)'\boldsymbol{X} + \boldsymbol{C}\boldsymbol{X}'d\boldsymbol{X}\right] &= \mathrm{Tr}\left(\boldsymbol{C}\left(d\boldsymbol{X}\right)'\boldsymbol{X}\right) + \mathrm{Tr}\left(\boldsymbol{C}\boldsymbol{X}'d\boldsymbol{X}\right) \\ &= \mathrm{Tr}\left[\boldsymbol{X}'\left(d\boldsymbol{X}\right)\boldsymbol{C}'\right] + \mathrm{Tr}\left(\boldsymbol{C}\boldsymbol{X}'d\boldsymbol{X}\right) \\ &= \mathrm{Tr}\left(\boldsymbol{C}'\boldsymbol{X}'d\boldsymbol{X}\right) + \mathrm{Tr}\left(\boldsymbol{C}\boldsymbol{X}'d\boldsymbol{X}\right) \\ &= \mathrm{Tr}\left[\left(\boldsymbol{C} + \boldsymbol{C}'\right)\boldsymbol{X}'d\boldsymbol{X}\right] \\ &= \mathrm{Tr}\left[\left(\boldsymbol{C} + \boldsymbol{C}'\right)'\boldsymbol{X}'d\boldsymbol{X}\right] \\ &= \mathrm{Tr}\left(\left[\boldsymbol{X}\left(\boldsymbol{C} + \boldsymbol{C}'\right)\right]'d\boldsymbol{X}\right) \end{aligned}$$

Using Equation (4.9), this becomes

$$\mathrm{Tr}\left(\left[\boldsymbol{X}\left(\boldsymbol{C} + \boldsymbol{C}'\right)\right]'d\boldsymbol{X}\right) = \left(\mathrm{vec}\left[\boldsymbol{X}\left(\boldsymbol{C} + \boldsymbol{C}'\right)\right]\right)'\mathrm{vec}\left(d\boldsymbol{X}\right).$$

Then $d\,\mathrm{Tr}\left(\boldsymbol{C}\boldsymbol{X}\right) = \mathrm{Tr}\left[\left(\boldsymbol{C} + \boldsymbol{C}'\right)\boldsymbol{X}'d\boldsymbol{X}\right]$ and $\dfrac{\partial\,\mathrm{Tr}\left(\boldsymbol{C}\boldsymbol{X}'\boldsymbol{X}\right)}{\partial\left[\mathrm{vec}\left(\boldsymbol{X}\right)\right]'} = \left(\mathrm{vec}\left[\boldsymbol{X}\left(\boldsymbol{C} + \boldsymbol{C}'\right)\right]\right)'$ by the First Identification Theorem. □

When the matrix argument $\boldsymbol{X}$ is symmetric, this imposes restrictions on the possible values that $\boldsymbol{X}$ may take, which in turn changes the form of the derivatives. The next two proofs address the derivative of the trace and the determinant when $\boldsymbol{X}$ is symmetric. Although similar results have been presented before (Harville, 1997, sections 15.6 and 15.8), these proofs using the duplication matrix are apparently new.

Lemma 4.24. *Let $\boldsymbol{X}$ be a $p \times p$ symmetric matrix and let $\boldsymbol{C}$ be $p \times p$ matrix. Let y be the scalar function defined by $y = \operatorname{Tr}(\boldsymbol{XC}) = \operatorname{Tr}(\boldsymbol{CX})$. Then*

$$dy = \operatorname{Tr}(\boldsymbol{C}d\boldsymbol{X})$$

and

$$\frac{\partial y}{\partial (\operatorname{vech}[\boldsymbol{X}])'} = (\operatorname{vech}[\boldsymbol{C} + \boldsymbol{C}' - \operatorname{diag}(\boldsymbol{C})])'.$$

Proof. Using Property (9) of the matrix differential,

$$\begin{aligned} dy &= d \operatorname{Tr}(\boldsymbol{CX}) \\ &= \operatorname{Tr}[d(\boldsymbol{CX})] \\ &= \operatorname{Tr}(\boldsymbol{C}d\boldsymbol{X}). \end{aligned}$$

To find the derivative, note that

$$\begin{aligned} dy &= d \operatorname{Tr}(\boldsymbol{XC}) \\ &= d \operatorname{Tr}(\boldsymbol{X}'\boldsymbol{C}) \\ &= \operatorname{Tr}[d(\boldsymbol{X}'\boldsymbol{C})] \\ &= \operatorname{Tr}[(d\boldsymbol{X}')\boldsymbol{C}] \\ &= \operatorname{Tr}[(d\boldsymbol{X})'\boldsymbol{C}] \\ &= \operatorname{vec}(d\boldsymbol{X})' \operatorname{vec}(\boldsymbol{C}) \end{aligned}$$

From Equation (4.12), this is equivalent to

$$\begin{aligned} dy &= \operatorname{vec}(d\boldsymbol{X})' \operatorname{vec}(\boldsymbol{C}) \\ &= \left[\boldsymbol{D}_p \operatorname{vech}(d\boldsymbol{X})\right]' \operatorname{vec}(\boldsymbol{C}) \\ &= [\operatorname{vech}(d\boldsymbol{X})]' \boldsymbol{D}_p' \operatorname{vec}(\boldsymbol{C}) \\ &= \left[\boldsymbol{D}_p' \operatorname{vec}(\boldsymbol{C})\right]' \operatorname{vech}(d\boldsymbol{X}) \\ &= (\operatorname{vech}[\boldsymbol{C} + \boldsymbol{C}' - \operatorname{diag}(\boldsymbol{C})])' \operatorname{vech}(d\boldsymbol{X}) \end{aligned}$$

where the next to the last equation follows because, since dy is a scalar, it is equal to its transpose. By the First Identification Theorem, $d\operatorname{Tr}(\boldsymbol{CX}) = \operatorname{Tr}(\boldsymbol{C}d\boldsymbol{X})$ and $\dfrac{\partial \operatorname{Tr}(\boldsymbol{CX})}{\partial [\operatorname{vech}(\boldsymbol{X})]'} = \left[\boldsymbol{D}_p' \operatorname{vec}(\boldsymbol{C})\right]' = (\operatorname{vech}[\boldsymbol{C} + \boldsymbol{C}' - \operatorname{diag}(\boldsymbol{C})])'$. □

Lemma 4.25. *Let $\boldsymbol{X}$ be a $p \times p$ symmetric matrix, and let y be the scalar function defined by $y = \log(|\boldsymbol{X}|)$. Then, at the points where $|\boldsymbol{X}| > 0$,*

$$dy = \operatorname{Tr}\left(\boldsymbol{X}^{-1} d\boldsymbol{X}\right)$$

and

$$\frac{\partial y}{\partial [\operatorname{vech}(\boldsymbol{X})]'} = \left(\operatorname{vech}\left[2\boldsymbol{X}^{-1} - \operatorname{diag}\left(\boldsymbol{X}^{-1}\right)\right]\right)'.$$

Proof. Noting that dy is a scalar and equal to its transpose and using Lemma 4.22,

$$\begin{aligned} dy &= \operatorname{Tr}\left(\boldsymbol{X}^{-1} d\boldsymbol{X}\right) \\ &= \left[\operatorname{vec}\left(\boldsymbol{X}^{-1\prime}\right)\right]' \operatorname{vec}(d\boldsymbol{X}) \\ &= \left[\operatorname{vec}\left(\boldsymbol{X}^{-1}\right)\right]' \operatorname{vec}(d\boldsymbol{X}) \\ &= [\operatorname{vec}(d\boldsymbol{X})]' \operatorname{vec}\left(\boldsymbol{X}^{-1}\right) \end{aligned}$$

From Equation (4.12), this is equivalent to

$$\begin{aligned} dy &= \left[\text{vec}\left(d\boldsymbol{X}\right)\right]' \text{vec}\left(\boldsymbol{X}^{-1}\right) \\ &= \left[\boldsymbol{D}_p \, \text{vech}\left(d\boldsymbol{X}\right)\right]' \text{vec}\left(\boldsymbol{X}^{-1}\right) \\ &= \left[\text{vech}\left(d\boldsymbol{X}\right)\right]' \boldsymbol{D}_p' \, \text{vec}\left(\boldsymbol{X}^{-1}\right) \\ &= \left[\boldsymbol{D}_p' \, \text{vec}\left(\boldsymbol{X}^{-1}\right)\right]' \text{vech}\left(d\boldsymbol{X}\right) \\ &= \left(\text{vech}\left[\boldsymbol{X}^{-1} + \boldsymbol{X}^{-1\prime} - \text{diag}\left(\boldsymbol{X}^{-1}\right)\right]\right)' \text{vech}\left(d\boldsymbol{X}\right) \\ &= \left(\text{vech}\left[2\boldsymbol{X}^{-1} - \text{diag}\left(\boldsymbol{X}^{-1}\right)\right]\right)' \text{vech}\left(d\boldsymbol{X}\right). \end{aligned}$$

So $d \log\left(|\boldsymbol{X}|\right) = \text{Tr}\left(\boldsymbol{X}^{-1}\right) d\boldsymbol{X}$ and $\dfrac{\partial \log\left(|\boldsymbol{X}|\right)}{\partial \left[\text{vech}\left(\boldsymbol{X}\right)\right]'} = \left[\boldsymbol{D}_p' \, \text{vec}\left(\boldsymbol{X}^{-1}\right)\right]' = \left(\text{vech}\left[2\boldsymbol{X}^{-1} - \text{diag}\left(\boldsymbol{X}^{-1}\right)\right]\right)'$ by the First Identification Theorem. □

4.3.2.2 Matrix Functions of a Matrix

Lemma 4.26. *Let $\boldsymbol{X}$, $\boldsymbol{C}_1$, and $\boldsymbol{C}_2$ be $p \times p$ matrices, and let $\boldsymbol{Y}$ be the $p \times p$ matrix defined by the transformation $\boldsymbol{Y} = \boldsymbol{C}_1 \boldsymbol{X} \boldsymbol{C}_2$. Then the Jacobian of the transformation is*

$$J_{\boldsymbol{X} \to \boldsymbol{Y}} = |\boldsymbol{C}_1|^p \, |\boldsymbol{C}_2|^p$$

Proof. This proof is outlined in Magnus and Neudecker (1980). Let

$$\boldsymbol{Y} = \boldsymbol{C}_1 \boldsymbol{X} \boldsymbol{C}_2$$

and take differentials of both sides to obtain

$$d\boldsymbol{Y} = \boldsymbol{C}_1 \left(d\boldsymbol{X}\right) \boldsymbol{C_2}.$$

Applying the vec operator gives

$$\operatorname{vec}(d\boldsymbol{Y}) = \operatorname{vec}(\boldsymbol{C}_1 (d\boldsymbol{X}) \boldsymbol{C}_2),$$

which is equivalent to

$$d\operatorname{vec}(\boldsymbol{Y}) = (\boldsymbol{C}_2' \otimes \boldsymbol{C}_1)\, d\operatorname{vec}(\boldsymbol{X}).$$

Thus

$$\begin{aligned} J_{\boldsymbol{X}\to\boldsymbol{Y}} &= \left|\boldsymbol{C}_2' \otimes \boldsymbol{C}_1\right| \\ &= \left|\boldsymbol{C}_2'\right|^p |\boldsymbol{C}_1|^p \\ &= |\boldsymbol{C}_1|^p |\boldsymbol{C}_2|^p \end{aligned}$$

□

Lemma 4.27. *Let $\boldsymbol{X}$ be a $p \times p$ symmetric matrix. Let $\boldsymbol{Y}$ be the $p \times p$ symmetric matrix defined by the transformation $\boldsymbol{Y} = \boldsymbol{X}^{-1}$. Then the Jacobian of the transformation is*

$$J_{\boldsymbol{X}\to\boldsymbol{Y}} = |\boldsymbol{X}|^{-(p+1)}$$

Proof. This proof is similar to that of Henderson and Searle (1979) and Magnus and Neudecker (1980). In this case, it is easier to find the Jacobian $J_{\boldsymbol{Y}\to\boldsymbol{X}}$ and apply Property (2) of Jacobians. Let

$$\boldsymbol{Y} = \boldsymbol{X}^{-1}$$

or, equivalently,

$$\boldsymbol{Y} = \boldsymbol{X}^{-1}\boldsymbol{X}\boldsymbol{X}^{-1}$$

Taking differentials yields

$$d\boldsymbol{Y} = \boldsymbol{X}^{-1}\left(d\boldsymbol{X}\right)\boldsymbol{X}^{-1}$$

and applying the vech operator gives

$$\operatorname{vech}\left(d\boldsymbol{Y}\right) = \operatorname{vech}\left(\boldsymbol{X}^{-1}\left(d\boldsymbol{X}\right)\boldsymbol{X}^{-1}\right).$$

This is equivalent to

$$d\operatorname{vech}\left(\boldsymbol{Y}\right) = -\boldsymbol{L}_p\left(\boldsymbol{X}^{-1}\otimes\boldsymbol{X}^{-1}\right)\boldsymbol{D}_p d\operatorname{vech}\left(\boldsymbol{X}\right).$$

Thus, using the First Identification Theorem,

$$J_{\boldsymbol{Y}\to\boldsymbol{X}} = \left|\boldsymbol{L}_p\left(\boldsymbol{X}^{-1}\otimes\boldsymbol{X}^{-1}\right)\boldsymbol{D}_p\right|.$$

Applying Lemma 4.15 gives

$$\begin{aligned}\left|\boldsymbol{L}_p\left(\boldsymbol{X}^{-1}\otimes\boldsymbol{X}^{-1}\right)\boldsymbol{D}_p\right| &= \left|\boldsymbol{X}^{-1}\right|^{p+1}\\ &= \left|\boldsymbol{X}\right|^{-(p+1)}.\end{aligned} \tag{4.25}$$

Then $J_{\boldsymbol{X}\to\boldsymbol{Y}} = 1/J_{\boldsymbol{Y}\to\boldsymbol{X}} = 1/\left|\boldsymbol{X}\right|^{-(p+1)} = \left|\boldsymbol{X}\right|^{p+1}$. □

Lemma 4.28. *Let $\boldsymbol{X}$ be a $p\times p$ symmetric matrix, and let $\boldsymbol{C}$ be a $p\times p$ matrix of constants. Let $\boldsymbol{Y}$ be the $p\times p$ symmetric matrix defined by the transformation $\boldsymbol{Y} = \boldsymbol{C}\boldsymbol{X}\boldsymbol{C}'$. Then the Jacobian of the transformation is*

$$J_{\boldsymbol{X}\to\boldsymbol{Y}} = \left|\boldsymbol{C}\right|^{-(p+1)}$$

Proof. This proof follows ones given in Magnus and Neudecker (1980, p. 436) and Henderson

and Searle (1979, p. 74). In this case, it is easier to find the Jacobian $J_{\boldsymbol{Y}\to\boldsymbol{X}}$ and apply Property (2) of Jacobians. Let

$$\boldsymbol{Y} = \boldsymbol{C}\boldsymbol{X}\boldsymbol{C}'$$

Taking differentials gives

$$d\boldsymbol{Y} = \boldsymbol{C}\,(d\boldsymbol{X})\,\boldsymbol{C}'$$

Applying the vech operator to both sides gives

$$\operatorname{vech}(d\boldsymbol{Y}) = \operatorname{vech}(\boldsymbol{C}\,(d\boldsymbol{X})\,\boldsymbol{C}')$$

or

$$d(\operatorname{vech}(\boldsymbol{Y})) = \boldsymbol{L}_p\,(\boldsymbol{C}\otimes\boldsymbol{C})\,\boldsymbol{D}_p\,d(\operatorname{vech}(\boldsymbol{X}))$$

using Equation (4.13). Thus, by the First Identification Theorem,

$$J_{\boldsymbol{Y}\to\boldsymbol{X}} = \left|\boldsymbol{L}_p\,(\boldsymbol{C}\otimes\boldsymbol{C})\,\boldsymbol{D}_p\right|,$$

which, by Lemma 4.15, is equal to $|\boldsymbol{C}|^{p+1}$. Thus $J_{\boldsymbol{X}\to\boldsymbol{Y}} = 1/J_{\boldsymbol{Y}\to\boldsymbol{X}} = 1/|\boldsymbol{C}|^{p+1} = |\boldsymbol{C}|^{-(p+1)}$. □

Lemma 4.29. *Let $\boldsymbol{X}$ be a $p\times p$ symmetric matrix, and let $\boldsymbol{Y}$ be the $p\times p$ symmetric matrix defined by the transformation $\boldsymbol{Y} = \boldsymbol{X}^{1/2}$. Then the Jacobian of the transformation is*

$$J_{\boldsymbol{X}\to\boldsymbol{Y}} = \prod_{i\le j}^{p}\left(\lambda_i + \lambda_j\right)$$

where $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_p$ are the ordered eigenvalues of $\boldsymbol{Y} = \boldsymbol{X}^{1/2}$.

Proof. A proof of a similar result is given in Magnus and Neudecker (1980, p. 437), while an

alternative derivation is given in Olkin and Sampson (1972, p. 269). Let

$$\boldsymbol{Y} = \boldsymbol{X}^{1/2} \Leftrightarrow \boldsymbol{X} = \boldsymbol{Y}^2 = \boldsymbol{Y}\boldsymbol{Y}.$$

Taking differentials of both sides using the product rule gives

$$d\boldsymbol{X} = \boldsymbol{Y}\,(d\boldsymbol{Y}) + (d\boldsymbol{Y})\,\boldsymbol{Y}.$$

Applying the vech operator gives

$$\begin{aligned} \operatorname{vech}(d\boldsymbol{X}) &= \operatorname{vech}(\boldsymbol{Y}\,(d\boldsymbol{Y}) + (d\boldsymbol{Y})\,\boldsymbol{Y}) \\ &= \operatorname{vech}(\boldsymbol{Y}\,(d\boldsymbol{Y})) + \operatorname{vech}((d\boldsymbol{Y})\,\boldsymbol{Y}) \end{aligned} \tag{4.26}$$

and using Equation (4.13) yields

$$\operatorname{vech}(d\boldsymbol{X}) = \boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{Y}\right)\boldsymbol{D}_p \operatorname{vech}(d\boldsymbol{Y}) + \boldsymbol{L}_p\left(\boldsymbol{Y} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p \operatorname{vech}(d\boldsymbol{Y})$$

or

$$d\operatorname{vech}(\boldsymbol{X}) = \boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{Y} + \boldsymbol{Y} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p\, d\operatorname{vech}(\boldsymbol{Y}). \tag{4.27}$$

By the First Identification Theorem, $J_{\boldsymbol{X}\to\boldsymbol{Y}} = \left|\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{Y} + \boldsymbol{Y} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p\right| = \prod_{i\le j}^{p}\left(\lambda_i + \lambda_j\right)$, where $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_p$ are the ordered eigenvalues of $\boldsymbol{Y} = \boldsymbol{X}^{1/2}$, by Lemma 4.16. □

Lemma 4.30. *Let $\boldsymbol{X}$ and $\boldsymbol{K}$ be $p \times p$ symmetric matrices, and let $\boldsymbol{Y}$ be the $p \times p$ symmetric matrix defined by the transformation $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{C}\boldsymbol{X}$. Then the Jacobian of the transformation is*

$$J_{\boldsymbol{X}\to\boldsymbol{Y}} = \frac{1}{\prod_{i\le j}^{p}\left(\eta_i + \eta_j\right)}$$

where $\eta_1 \geq \eta_2 \geq \cdots \geq \eta_p$ are the ordered eigenvalues of $\boldsymbol{XC}$.

Proof. Similar proofs are given in Magnus and Neudecker (1980, p. 437), with Olkin and Sampson (1972, p. 269) providing another approach. In this case, it is easier to find the Jacobian $J_{\boldsymbol{Y}\to\boldsymbol{X}}$ and apply Property (2) of Jacobians. Let

$$\boldsymbol{Y} = \boldsymbol{XCX}$$

Taking differentials of both sides using the product rule gives

$$d\boldsymbol{Y} = \boldsymbol{XC}\,(d\boldsymbol{X}) + (d\boldsymbol{X})\,\boldsymbol{CX}.$$

Applying the vech operator gives

$$\begin{aligned}\operatorname{vech}(d\boldsymbol{Y}) &= \operatorname{vech}(\boldsymbol{XC}\,(d\boldsymbol{X}) + (d\boldsymbol{X})\,\boldsymbol{CX}) \\ &= \operatorname{vech}(\boldsymbol{XC}\,(d\boldsymbol{X})) + \operatorname{vech}((d\boldsymbol{X})\,\boldsymbol{CX})\end{aligned} \tag{4.28}$$

and using Equation (4.13) yields

$$\begin{aligned}\operatorname{vech}(d\boldsymbol{Y}) &= \boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{XC}\right)\boldsymbol{D}_p \operatorname{vech}(d\boldsymbol{X}) + \boldsymbol{L}_p\left((\boldsymbol{CX})' \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p \operatorname{vech}(d\boldsymbol{X}) \\ &= \boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{XC}\right)\boldsymbol{D}_p \operatorname{vech}(d\boldsymbol{X}) + \boldsymbol{L}_p\left(\boldsymbol{X}'\boldsymbol{C}' \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p \operatorname{vech}(d\boldsymbol{X}) \\ &= \boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{XC}\right)\boldsymbol{D}_p \operatorname{vech}(d\boldsymbol{X}) + \boldsymbol{L}_p\left(\boldsymbol{XC} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p \operatorname{vech}(d\boldsymbol{X})\end{aligned}$$

since $\boldsymbol{C}$ and $\boldsymbol{X}$ are symmetric. This simplifies to

$$d(\operatorname{vech}(\boldsymbol{Y})) = \boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{XC} + \boldsymbol{XC} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p\, d(\operatorname{vech}(\boldsymbol{X})) \tag{4.29}$$

By the First Identification Theorem, $J_{\boldsymbol{Y}\to\boldsymbol{X}} = \left|\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{XC} + \boldsymbol{XC} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p\right| = \prod_{i\leq j}^{p}\left(\eta_i + \eta_j\right)$, where $\eta_1 \geq \eta_2 \geq \cdots \geq \eta_p$ are the ordered eigenvalues of $\boldsymbol{XC}$, by Lemma 4.16. Thus $J_{\boldsymbol{X}\to\boldsymbol{Y}} =$

$1/J_{\boldsymbol{Y}\to\boldsymbol{X}} = 1/\prod_{i\le j}^{p}\left(\eta_i+\eta_j\right)$. □

Lemma 4.31. *Let $\boldsymbol{X}$ be a $p\times p$ lower triangular matrix. Let $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ be $p\times p$ symmetric matrices, and let $\boldsymbol{Y}$ be the $p\times p$ symmetric matrix defined by $\boldsymbol{Y}=\boldsymbol{X}'\boldsymbol{C}_1\boldsymbol{X}+\boldsymbol{X}\boldsymbol{C}_2\boldsymbol{X}'$. Then the Jacobian of the transformation is*

$$J_{\boldsymbol{X}\to\boldsymbol{Y}} = 2^p\left|\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{C}_1\boldsymbol{X}+\boldsymbol{C}_2\boldsymbol{X}'\otimes\boldsymbol{I}_p\right)\boldsymbol{D}_p\right|.$$

One specific case is the Cholesky decomposition $\boldsymbol{Y}=\boldsymbol{X}\boldsymbol{X}'$, with the Jacobian

$$\begin{aligned} J_{\boldsymbol{X}\to\boldsymbol{Y}} &= 2^n\left|\boldsymbol{L}_p\left(\boldsymbol{X}\otimes\boldsymbol{I}_p\right)\boldsymbol{D}_p\right| \\ &= 2^p\prod_{i=1}^{p}x_{ii}^{p-i+1}, \end{aligned}$$

where x_{ii} are the diagonal elements of $\boldsymbol{X}$. A second specific case is the transformation $\boldsymbol{Y}=\boldsymbol{X}'\boldsymbol{C}\boldsymbol{X}$, with the Jacobian

$$\begin{aligned} J_{\boldsymbol{X}\to\boldsymbol{Y}} &= 2^n\left|\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{C}\boldsymbol{X}\right)\boldsymbol{D}_p\right| \\ &= 2^p\prod_{i=1}^{p}\left(x_{ii}^{i}\left|\boldsymbol{C}_{[i]}\right|\right), \end{aligned}$$

where $\boldsymbol{C}^{[i]}$ denotes the $i\times i$ upper left submatrix of $\boldsymbol{C}$ and $\boldsymbol{C}_{[i]}$ denotes the $i\times i$ lower right submatrix of $\boldsymbol{C}$.

Lemma 4.32. *Let $\boldsymbol{X}$ be a $p_1\times p_1$ singular matrix of rank k. Let $\boldsymbol{Q}$ be a $p_2\times p_2$ matrix of full rank, and let $\boldsymbol{Y}$ be the $p_2\times p_2$ matrix of rank k given by $\boldsymbol{Y}=\boldsymbol{Q}\boldsymbol{X}\boldsymbol{Q}'$. Let the spectral decomposition of $\boldsymbol{X}$ be denoted as $\boldsymbol{X}=\boldsymbol{E}_1\boldsymbol{L}_1\boldsymbol{E}_1$, so that $\boldsymbol{E}_1'\boldsymbol{E}_1=\boldsymbol{I}_p$ and $\boldsymbol{L}_1=\operatorname{diag}(\lambda_1,\lambda_2,\ldots,\lambda_k)$ are the ordered nonzero eigenvalues of $\boldsymbol{X}$. Similarly, let the spectral decomposition of $\boldsymbol{Y}$ be denoted as $\boldsymbol{Y}=\boldsymbol{E}_2\boldsymbol{L}_2\boldsymbol{E}_2'$, so that $\boldsymbol{E}_2'\boldsymbol{E}_2=\boldsymbol{I}_k$ and $\boldsymbol{L}_2=\operatorname{diag}(\kappa_1\ge\kappa_2\ge\cdots\ge\kappa_k)$ are the ordered nonzero eigenvalues of $\boldsymbol{Y}$. Then the Jacobian of the transformation $\boldsymbol{Y}=\boldsymbol{F}(\boldsymbol{X})$ is*

$$J_{\boldsymbol{X}\to\boldsymbol{Y}} = |\boldsymbol{L}_1|^{(k-p_2-1)/2}\,|\boldsymbol{L}_2|^{(p_2+1-k)/2}\,|\boldsymbol{Q}|^{-k}$$

Proof. This is a more specific statement of Theorem 1 of Díaz-García and Gutiérrez Jáimez (1997) and its accompanying proof. Note that the original article contains a typographical error, as the exponent for the first determinant should be $k - p_2 - 1$ rather than $p_2 + 1 - k$. □

Lemma 4.33. *Let $\boldsymbol{X}$ be a $p \times p$ positive semidefinite matrix of rank k, and let $\boldsymbol{Y}$ be the positive semidefinite $p \times p$ matrix of rank k given by $\boldsymbol{Y} = \boldsymbol{X}^{1/2}$. Let the spectral decomposition of $\boldsymbol{X}$ be denoted by $\boldsymbol{X} = \boldsymbol{ELE}'$, so that $\boldsymbol{E}'\boldsymbol{E} = \boldsymbol{I}_p$ and $\boldsymbol{L} = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_k)$ are the ordered nonzero eigenvalues of $\boldsymbol{X}$. Then the Jacobian of the transformation from $\boldsymbol{X}$ to $\boldsymbol{Y}$ is given by*

$$J_{\boldsymbol{X} \to \boldsymbol{Y}} = \prod_{i \leq j}^{k} \left(\lambda_i + \lambda_j\right) = 2^k \left|\boldsymbol{L}\right|^{p-k+1} \prod_{i<j}^{k} \left(\lambda_i + \lambda_j\right). \tag{4.30}$$

Proof. This result is proved as Theorem 8 of Díaz-García and González-Farías (2005). □

4.3.3 Probability Distributions

The development of the multivariate RVM requires several results from multivariate distribution theory. This theory extends the well-known univariate statistical distributions to multivariate data, in which multiple data points are collected on each experimental unit. The multivariate distributions allow for modeling more complicated covariance structures among the outcome variables, as data points collected from a single experimental unit are often presumed to be correlated. The next three sections review the key distributions in multivariate statistical analysis. Of necessity, only certain aspects of distribution theory germane to the multivariate RVM will be presented; a comprehensive treatment of the subject may be found in Gupta and Nagar (2000). Proofs for many of these results are available in the literature and will be appropriately referenced. However, proofs will also be restated as needed, which will provide additional details not contained in extant works and illustrate techniques that will be used in the development of new distributions in the present work.

4.3.3.1 Wishart and Related Distributions

The Wishart distribution (Wishart, 1928) is a multivariate generalization of the univariate χ^2 distribution, which is in turn a special case of the univariate gamma distribution given by Equation (3.2). The discovery and characterization of the Wishart distribution was a key event in the development of multivariate analysis. A large body of literature now exists that examines aspects of the Wishart, and a number of related distributions have been described. The formal definition of the Wishart distribution is given by the following.

Definition 4.34 (Wishart Distribution). *The $p \times p$ positive definite matrix $\boldsymbol{M}$ has a Wishart distribution with $n \geq p$ degrees of freedom and positive definite covariance matrix $\boldsymbol{\Sigma}$, denoted as $\boldsymbol{M} \sim W_p(\boldsymbol{\Sigma}, m)$, if the probability density of $\boldsymbol{M}$ is*

$$f(\boldsymbol{M}) = \frac{|\boldsymbol{M}|^{(n-p-1)/2}}{2^{np/2}\,|\boldsymbol{\Sigma}|^{n/2}\,\Gamma_p\left(\frac{n}{2}\right)} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}\boldsymbol{M}\right]\right) \tag{4.31}$$

where $\Gamma_p(\cdot)$ is the multivariate gamma function

$$\begin{aligned}\Gamma_p(x) &= \pi^{p(p-1)/4} \prod_{i=1}^{p} \Gamma\left(x - \tfrac{1}{2}(i-1)\right), \qquad x > \tfrac{1}{2}(p-1) \\ &= \int_{\boldsymbol{X}>0} e^{-\operatorname{Tr}\boldsymbol{X}}\,|\boldsymbol{X}|^{x-(m+1)/2}\,\mathrm{d}\boldsymbol{X}\end{aligned}$$

with $\boldsymbol{X} > 0$ denoting the space of all positive definite matrices.

The Wishart is the sampling distribution of the sample covariance matrix for multivariate data that are normally distributed. Specifically, if $\boldsymbol{x}_j \sim N_p(\boldsymbol{0}, \boldsymbol{\Sigma})$, $j = 1, 2, \ldots, n$ represent $n \geq p$ observations from a multivariate normal population with mean $\boldsymbol{0}$ and covariance matrix $\boldsymbol{\Sigma}$, then

$$\boldsymbol{M} = \sum_{j=1}^{n} \boldsymbol{x}_j \boldsymbol{x}_j'$$

has a p-variate Wishart distribution with n degrees of freedom.

When $p = 1$, the Wishart distribution reduces to $W_1\left(\sigma^2, n\right) \equiv \sigma^2\,\mathrm{Gamma}\left(\frac{n}{2}, \frac{1}{2}\right) \equiv \sigma^2\chi^2_n$,

where $\underset{1\times1}{\Sigma} = \sigma^2$. In general, the diagonal elements m_{ii} of a Wishart-distributed matrix $\boldsymbol{M}$ are proportional to a χ^2 distribution. Letting σ_{ii}^2 denote the ith diagonal element of $\boldsymbol{\Sigma}$, this relationship can be expressed as $m_{ii} \sim \sigma_{ii}^2 \chi_n^2$. In Bayesian analysis, a Wishart prior is the usual choice for the covariance matrix of the multivariate normal distribution, analogous to the gamma prior for the variance in the univariate normal.

The matrix $\boldsymbol{M}$ in the above definition is positive definite with probability 1 if and only if $n \geq p$ (Dykstra, 1970; Eaton and Perlman, 1973). Thus, if $n < p$, so that the matrix $\boldsymbol{M}$ is singular, the distribution in Equation (4.31) does not exist. However, a related distribution can be defined for such singular matrices, which is given next.

Definition 4.35 (**Pseudo-Wishart Distribution**). *The $p \times p$ positive semidefinite matrix $\boldsymbol{M}$ of rank $n < p$ has a pseudo-Wishart distribution with n degrees of freedom and covariance matrix $\boldsymbol{\Sigma}$, denoted as $\boldsymbol{M} \sim W_p^n(\boldsymbol{\Sigma}, n)$, if the probability density of $\boldsymbol{M}$ is*

$$f(\boldsymbol{M}) = \frac{\pi^{\left(-pn+p^2\right)}}{2^{pn/2}\left|\boldsymbol{\Sigma}\right|^{n/2}\Gamma_n\left(\frac{n}{2}\right)}\left|\boldsymbol{L}\right|^{(n-p-1)/2}\exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}\boldsymbol{M}\right]\right) \tag{4.32}$$

where $\boldsymbol{L} = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$, with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ being the ordered nonzero eigenvalues of $\boldsymbol{M}$.

The nomenclature for this distribution is not standardized. Srivastava and Khatri (1979, p.72) and Díaz-García *et al.* (1997) suggest the term *singular* Wishart distribution be used when the population covariance matrix $\boldsymbol{\Sigma}$ is singular, and that the term *pseudo*-Wishart distribution be used when the sample matrix $\boldsymbol{M}$ is singular, which will be the convention used in the present work. However, this practice has not been widely adopted, and the distribution in Equation (4.32) is frequently labeled the singular Wishart distribution in the literature. The pseudo-Wishart distribution is an analogue of the nonsingular Wishart distribution that describes the behavior of multivariate normal data when the number of observations n is less than the number of observed variables p. Specifically, if $\boldsymbol{x}_j \sim N_p(0, \boldsymbol{\Sigma})$, $j = 1, 2, \ldots, n$ represent $n < p$ observations from a multivariate

normal population with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$, then

$$\boldsymbol{M} = \sum_{j=1}^{n} \boldsymbol{x}_j \boldsymbol{x}_j'$$

has a p-variate pseudo-Wishart distribution with n degrees of freedom. Interest in the pseudo-Wishart and related multivariate singular distributions has grown recently due to the need to analyze high-dimensional data, such as occurs with microarrays (Srivastava, 2007). The distributional theory and applications of the pseudo-Wishart distribution are described in detail by Uhlig (1994), Díaz-García *et al.* (1997), and Srivastava (2003).

Comparing Equations (4.31) and (4.32), it is readily apparent that the two distributions are equivalent for $n = p$. However, this equivalence does not hold in general, as shown in the following lemma, which appears to be new to the literature.

Lemma 4.36. *If $n \geq p + 1$, then the relationship between the nonsingular Wishart distribution in Equation* (4.31) *and the singular Wishart distribution in Equation* (4.32) *is given by*

$$W_p^n(\boldsymbol{\Sigma}, n) = \frac{\pi^{(n-p)^2/2}}{\Gamma_{n-p}\left(\frac{n-p}{2}\right)} W_p(\boldsymbol{\Sigma}, n) \tag{4.33}$$

Proof. If $n \geq p$, Equation (4.32) becomes

$$\frac{\pi^{\left(-np+n^2\right)/2}}{2^{np/2}\Gamma_n\left(\frac{n}{2}\right)|\boldsymbol{\Sigma}|^{n/2}} |\boldsymbol{L}|^{(n-p-1)/2} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}\boldsymbol{M}\right]\right), \tag{4.34}$$

adopting the convention that $\boldsymbol{L} = \operatorname{diag}\left(l_1, l_2, \ldots, l_{min(n,p)}\right)$ is the diagonal matrix of the ordered nonzero eigenvalues of $\boldsymbol{M}$ whether $\boldsymbol{M}$ is singular or nonsingular. Comparing the Wishart distribution in Equation (4.31) to Equation (4.34), it can be seen that Equation (4.33) is true if

$$\frac{\pi^{\left(-np+n^2\right)/2}}{\Gamma_n\left(\frac{n}{2}\right)} = \frac{\pi^{(n-p)^2/2}}{\Gamma_{n-p}\left(\frac{n-p}{2}\right)} \cdot \frac{1}{\Gamma_p\left(\frac{n}{2}\right)}$$

or

$$\pi^{(-np+n^2)/2}\Gamma_p\left(\frac{n}{2}\right) = \frac{\pi^{(n-p)^2/2}}{\Gamma_{n-p}\left(\frac{n-p}{2}\right)} \cdot \Gamma_n\left(\frac{n}{2}\right) \tag{4.35}$$

Proof of the equality in Equation (4.35) proceeds by induction. Note that, by the definition of the multivariate gamma function,

$$\Gamma_n\left(\frac{n}{2}\right) = \pi^{n(n-1)/4} \prod_{j=1}^{n} \Gamma\left(\frac{n+1-j}{2}\right)$$

Furthermore, if $n > p$,

$$\begin{aligned}\Gamma_n\left(\frac{n}{2}\right) &= \pi^{n(n-1)/4} \prod_{j=1}^{n} \Gamma\left(\frac{n+1-j}{2}\right) \\ &= \pi^{p(p-1)/4} \cdot \prod_{j=1}^{p} \Gamma\left(\frac{n+1-j}{2}\right) \pi^{(n-p)(n-p-1)/4} \cdot \prod_{j=p+1}^{n} \Gamma\left(\frac{n+1-j}{2}\right) \cdot \pi^{p(n-p)/2}\end{aligned}$$

With a change of index on the second product term, this can be rewritten as

$$\Gamma_n\left(\frac{n}{2}\right) = \pi^{p(p-1)/4} \cdot \prod_{j=1}^{p} \Gamma\left(\frac{n+1-j}{2}\right) \pi^{(n-p)(n-p-1)/4} \cdot \prod_{j=1}^{n-p} \Gamma\left(\frac{n-p+1-j}{2}\right) \cdot \pi^{p(n-p)/2}$$

so that, using the definition of the multivariate gamma function,

$$\Gamma_n\left(\frac{n}{2}\right) = \Gamma_p\left(\frac{n}{2}\right)\Gamma_{n-p}\left(\frac{n-p}{2}\right)\pi^{p(n-p)/2} \tag{4.36}$$

For the base case of $n = p + 1$, Equation (4.35) becomes

$$\pi^{\left(-(p+1)p+(p+1)^2\right)/2}\Gamma_p\left(\frac{p+1}{2}\right) = \frac{\pi^{((p+1)-p)^2/2}}{\Gamma_{(p+1)-p}\left(\frac{(p+1)-p}{2}\right)} \cdot \Gamma_{p+1}\left(\frac{p+1}{2}\right)$$

or

$$\pi^{(p+1)/2}\Gamma_p\left(\frac{p+1}{2}\right) = \frac{\pi^{1/2}}{\Gamma_1\left(\frac{1}{2}\right)} \cdot \Gamma_{p+1}\left(\frac{p+1}{2}\right).$$

Since $\Gamma_1\left(\frac{1}{2}\right) = \Gamma\left(\frac{1}{2}\right) = \pi^{1/2}$, this can be written as

$$\pi^{(p+1)/2}\Gamma_p\left(\frac{p+1}{2}\right) = \Gamma_{p+1}\left(\frac{p+1}{2}\right). \tag{4.37}$$

Applying Equation (4.36) to the term on the right-hand side yields

$$\Gamma_{p+1}\left(\frac{p+1}{2}\right) = \Gamma_p\left(\frac{p+1}{2}\right)\Gamma_1\left(\frac{1}{2}\right)\pi^{p/2}$$

so that Equation (4.37) becomes

$$\begin{aligned}\pi^{(p+1)/2}\Gamma_p\left(\frac{p+1}{2}\right) &= \Gamma_p\left(\frac{p+1}{2}\right)\Gamma_1\left(\frac{1}{2}\right)\pi^{p/2} \\ &= \Gamma_p\left(\frac{p+1}{2}\right)\pi^{1/2}\pi^{p/2}.\end{aligned}$$

Thus Equation (4.35) holds for the base case. It must also be shown that, if equality holds for an arbitrary $n > p+1$, then equality holds for $n+1$. For the latter, Equation (4.35) becomes

$$\pi^{\left(-(n+1)p+(n+1)^2\right)/2}\Gamma_p\left(\frac{n+1}{2}\right) = \frac{\pi^{((n+1)-p)^2/2}}{\Gamma_{(n+1)-p}\left(\frac{(n+1)-p}{2}\right)} \cdot \Gamma_{n+1}\left(\frac{n+1}{2}\right)$$

This simplifies to

$$\pi^{\left(-np-p+n^2+2n+1\right)/2}\Gamma_p\left(\frac{n+1}{2}\right) = \frac{\pi^{\left(n^2+2n+1+p^2-2p-2np\right)/2}}{\Gamma_{n+1-p}\left(\frac{n+1-p}{2}\right)} \cdot \Gamma_{n+1}\left(\frac{n+1}{2}\right)$$

or

$$\Gamma_p\left(\frac{n+1}{2}\right) = \frac{\pi^{(p^2-p-np)/2}}{\Gamma_{n+1-p}\left(\frac{n+1-p}{2}\right)} \cdot \Gamma_{n+1}\left(\frac{n+1}{2}\right).$$

Substituting Equation (4.36) on the right-hand side gives gives

$$\Gamma_p\left(\frac{n+1}{2}\right) = \frac{\pi^{(p^2-p-np)/2}}{\Gamma_{n+1-p}\left(\frac{n+1-p}{2}\right)} \cdot \Gamma_p\left(\frac{n+1}{2}\right)\Gamma_{n+1-p}\left(\frac{n+1-p}{2}\right)\pi^{p(n+1-p)/2}$$

which becomes

$$1 = \pi^{(p^2-p-np)/2} \cdot \pi^{(np+p-p^2)/2}$$

Thus Equation (4.33) holds by induction. □

Based on Equation (4.33), the formulae for the nonsingular Wishart and the pseudo-Wishart distributions are equivalent only when $n = p$ or $n = p + 1$. As equivalence of the two does not occur in general, determining whether the sample matrix is singular will be essential prior to applying the multivariate RVM method. Separate computational routines will be required for the nonsingular and pseudo-Wishart distributions so that appropriate results may be obtained.

A final distribution related to the Wishart describes the distribution of the inverse of a Wishart-distributed random variable.

Definition 4.37 (**Inverted Wishart Distribution**). *The $p \times p$ positive definite matrix $\boldsymbol{M}$ has an inverted Wishart distribution with $n \geq p$ degrees of freedom and positive definite covariance matrix $\boldsymbol{\Sigma}$, denoted as $\boldsymbol{M} \sim IW_p(\boldsymbol{\Sigma}, m)$, if the probability density of $\boldsymbol{M}$ is*

$$f(\boldsymbol{M}) = \frac{|\boldsymbol{\Sigma}|^{(n-p-1)/2}}{2^{(n-p-1)p/2}\,|\boldsymbol{M}|^{n/2}\,\Gamma_p\left(\frac{n-p-1}{2}\right)} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{M}^{-1}\boldsymbol{\Sigma}\right]\right) \tag{4.38}$$

The inverted Wishart is frequently used in Bayesian analysis as the conjugate prior for the covariance matrix of the multivariate normal distribution. The Wishart and inverted Wishart dis-

tributions are closely related: if $\boldsymbol{M} \sim IW_p(\boldsymbol{\Sigma}, n)$, then $\boldsymbol{M}^{-1} \sim W_p\left(\boldsymbol{\Sigma}^{-1}, n-p-1\right)$.

4.3.3.2 Multivariate Beta and Related Distributions

A second family of distributions of great importance in multivariate analysis is the multivariate beta, which extends the univariate beta family of distributions. Specifically, the univariate random variable x is said to have a beta type I distribution with parameters a and b if the density function $f(x)$ is given by

$$f(x) = \frac{\Gamma\left(\frac{a+b}{2}\right)}{\Gamma\left(\frac{a}{2}\right)\Gamma\left(\frac{b}{2}\right)} x^{a/2-1}(1-x)^{b/2-1}, \quad 0 < x < 1. \tag{4.39}$$

The univariate random variable x is said to have a beta type II distribution with parameters a and b if the density function $f(x)$ is given by

$$f(x) = \frac{\Gamma\left(\frac{a+b}{2}\right)}{\Gamma\left(\frac{a}{2}\right)\Gamma\left(\frac{b}{2}\right)} x^{a/2-1}(1+x)^{-(a+b)/2}. \tag{4.40}$$

The beta type II distribution is related to the univariate F distribution with parameters a and b, which has the density function

$$f(x) = \frac{\Gamma\left(\frac{a+b}{2}\right)}{\Gamma\left(\frac{a}{2}\right)\Gamma\left(\frac{b}{2}\right)} \left(\frac{a}{b}\right)^{a/2} x^{a/2-1}\left(1+\frac{a}{b}x\right)^{-(a+b)/2}. \tag{4.41}$$

Also, note that if x has a beta type I distribution, then $\frac{x}{1-x}$ has a beta type II distribution, which has led some authors to call the latter the inverted beta distribution.

In univariate analysis, if $x_1 \sim \chi^2_{n_1}$ and $x_2 \sim \chi^2_{n_2}$ are independent random variables, then $\dfrac{x_1}{x_1+x_2}$ has a beta type I distribution. Similarly, $\dfrac{x_1}{x_2}$ has a beta type II distribution, and $\dfrac{x_1/n_1}{x_2/n_2}$ has an F_{n_1,n_2} distribution. The corresponding multivariate generalizations involve the ratios of determinants of Wishart distributed random variables. Several different approaches for generalizing the univariate case have been developed by different authors, which are summarized in the following definitions. The nomenclature for these distributions is also not standardized, with some authors describing

them as *multivariate* and others calling them *matrix variate*. The former will be used throughout the present work.

Definition 4.38 (**Multivariate Beta Distribution, Type I**). *The $p \times p$ positive definite, symmetric random matrix $\boldsymbol{M}$ has a multivariate Beta type I distribution with n_1 and n_2 degrees of freedom, denoted as $\boldsymbol{M} \sim M\beta_I(p, n_1, n_2)$, if the probability density of $\boldsymbol{M}$ is*

$$\frac{\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_p\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\boldsymbol{M}\right|^{(n_1-p-1)/2}\left|\boldsymbol{I}_p-\boldsymbol{M}\right|^{(n_2-p-1)/2}, \quad 0<\boldsymbol{M}<\boldsymbol{I}_p.$$

The multivariate beta type I distribution was originally described by Hsu (1939), with extensions by Khatri (1959, 1970) and Mitra (1970). It is a generalization of the univariate beta type I distribution. Let $\boldsymbol{M}_1 \sim W_p\left(n_1, \boldsymbol{I}_p\right)$ and $\boldsymbol{M}_2 \sim W_p\left(n_2, \boldsymbol{I}_p\right)$, and let $\left(\boldsymbol{M}_1+\boldsymbol{M}_2\right)^*$ be any factorization of $\boldsymbol{M}_1+\boldsymbol{M}_2$ such that $\left(\boldsymbol{M}_1+\boldsymbol{M}_2\right)^*\left(\boldsymbol{M}_1+\boldsymbol{M}_2\right)^{*\prime}=\boldsymbol{M}_1+\boldsymbol{M}_2$. Then $\left(\boldsymbol{M}_1+\boldsymbol{M}_2\right)^*\boldsymbol{M}_1\left(\boldsymbol{M}_1+\boldsymbol{M}_2\right)^{*\prime} \sim M\beta_I(p, n_1, n_2)$. This distribution is described here for completeness but will not be used in the derivation of the multivariate RVM.

Definition 4.39 (**Multivariate Beta Distribution, Type II**). *The $p \times p$ positive definite, symmetric random matrix $\boldsymbol{M}$ has a multivariate Beta type II distribution with n_1 and n_2 degrees of freedom, denoted as $\boldsymbol{M} \sim M\beta_{II}(p, n_1, n_2)$, if the probability density of $\boldsymbol{M}$ is*

$$\frac{\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_p\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\boldsymbol{M}\right|^{(n_1-p-1)/2}\left|\boldsymbol{I}_p+\boldsymbol{M}\right|^{-(n_1+n_2)/2}$$

The multivariate beta type II distribution was derived by Olkin and Rubin (1964). It is a generalization of the univariate beta type II distribution. Let the matrices $\boldsymbol{M}_1 \sim W_p\left(I_p, n_1\right)$ and $\boldsymbol{M}_2 \sim W_p\left(I_p, n_2\right)$ be independent; then $\boldsymbol{M}_1^{-1/2}\boldsymbol{M}_2\boldsymbol{M}_1^{-1/2\prime} \sim M\beta_{II}(p, n_1, n_2)$. When $p=1$, the multivariate beta type II distribution is equivalent to the univariate beta type II distribution. Also note that some authors, such as Srivastava and Khatri (1979, section 3.6), list the degrees of freedom for the multivariate type II distribution in reverse order compared to the above definition.

The preceding definition is occasionally called the *standardized* multivariate beta type II distribution, as the Wishart matrices from which it is derived are in standard form with covariance

matrix $\boldsymbol{I}_p$. The generalized multivariate beta type II distribution is an extension of the standardized distribution, given in the following definition.

Definition 4.40 (**Generalized Multivariate Beta Distribution, Type II**). *The $p \times p$ positive definite, symmetric random matrix $\boldsymbol{M}$ has a generalized multivariate Beta type II distribution with n_1 and n_2 degrees of freedom and parameter matrices $\boldsymbol{\Omega}$ and $\boldsymbol{\Xi}$, denoted $\boldsymbol{M} \sim G\beta_{II}(p, n_1, n_2; \boldsymbol{\Omega}, \boldsymbol{\Xi})$, if the probability density of $\boldsymbol{M}$ is*

$$\frac{\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_p\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\boldsymbol{\Omega}+\boldsymbol{\Psi}\right|^{n_2/2}\left|\boldsymbol{M}-\boldsymbol{\Xi}\right|^{(n_1-p-1)/2}\left|\boldsymbol{\Omega}+\boldsymbol{M}\right|^{-(n_1+n_2)/2}$$

The generalized multivariate beta type II distribution was derived by Tan (1969). It readily follows from the standardized distribution: if $\boldsymbol{M} \sim M\beta_{II}(p, n_1, n_2)$, then $(\boldsymbol{\Omega}+\boldsymbol{\Xi})^{1/2}\boldsymbol{M}(\boldsymbol{\Omega}+\boldsymbol{\Xi})^{1/2} + \boldsymbol{\Xi} \sim G\beta_{II}(p, n_1, n_2; \boldsymbol{\Omega}, \boldsymbol{\Xi})$. The matrix $\boldsymbol{\Xi}$ serves as a location parameter, while the matrix $\boldsymbol{\Omega}$ is a scale parameter.

The multivariate beta type II distribution is occasionally referred to as the multivariate F distribution, similar to the relationship between the univariate beta type II and univariate F distributions. However, an alternative formulation of the multivariate F distribution, which more closely resembles the definition of the univariate F distribution, has also been proposed.

Definition 4.41 (**Multivariate F Distribution**). *The $p \times p$ positive definite, symmetric random matrix $\boldsymbol{M}$ has a multivariate F-distribution with $n_1 \geq p$ and $n_2 \geq p$ degrees of freedom and scale matrix $\boldsymbol{\Omega}$, denoted as $\boldsymbol{M} \sim F_p(n_1, n_2, \boldsymbol{\Omega})$, if the probability density of $\boldsymbol{M}$ is*

$$\frac{\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_p\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\boldsymbol{\Omega}\right|^{-n_1/2}\left|\boldsymbol{M}\right|^{(n_1-p-1)/2}\left|\boldsymbol{I}_p+\boldsymbol{\Omega}^{-1}\boldsymbol{M}\right|^{-(n_1+n_2)/2}$$

This distribution is derived and extensively discussed in Muirhead (1982, section 8.2.5) and Muirhead and Verathaworn (1985), although the distribution was not specifically labeled until later publications (Konno, 1991). When $p = 1$, $F_1(n_1, n_2, \boldsymbol{\Omega}) \equiv F_{n_1,n_2}$ with $\boldsymbol{\Omega} = \dfrac{n_2}{n_1}$. More generally, if the matrix $\boldsymbol{M}_1 \sim W_p(\boldsymbol{\Omega}, n_1)$ and $\boldsymbol{M}_2 \sim W_p\left(\boldsymbol{I}_p, n_2\right)$, then $\boldsymbol{M}_1^{1/2}\boldsymbol{M}_2^{-1}\boldsymbol{M}_1^{1/2} \sim F_p(n_1, n_2, \boldsymbol{\Omega})$.

Since

$$\begin{aligned}
&\frac{\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_p\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\boldsymbol{\Omega}\right|^{n_2/2}\left|\boldsymbol{M}\right|^{(n_1-p-1)/2}\left|\boldsymbol{\Omega}+\boldsymbol{M}\right|^{-(n_1+n_2)/2} \\
&\qquad=\frac{\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_p\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\boldsymbol{\Omega}\right|^{n_2/2}\left|\boldsymbol{M}\right|^{(n_1-p-1)/2}\left|\boldsymbol{\Omega}+\boldsymbol{M}\right|^{-(n_1+n_2)/2}\cdot\left|\boldsymbol{\Omega}^{-1}\right|^{-(n_1+n_2)/2} \\
&\qquad\quad\cdot\left|\boldsymbol{\Omega}\right|^{-(n_1+n_2)/2} \\
&\qquad=\frac{\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_p\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\boldsymbol{\Omega}\right|^{-n_1/2}\left|\boldsymbol{M}\right|^{(n_1-p-1)/2}\left|\boldsymbol{I}_p+\boldsymbol{\Omega}^{-1}\boldsymbol{M}\right|^{-(n_1+n_2)/2},
\end{aligned}$$

it follows that $F_p\left(n_1,n_2,\boldsymbol{\Omega}\right)\equiv G\beta_{II}\left(p,n_1,n_2;\boldsymbol{\Omega},\boldsymbol{0}\right)$.

Olkin and Rubin (1964) also derived other distributions related to the multivariate beta type II by examining transformations of the product of Wishart densities. Although not as widely used as the multivariate beta type II, these distributions are of importance to the development of the multivariate RVM and are presented in the following theorems.

Theorem 4.42. *Let $\boldsymbol{M}_1\sim W_p\left(\boldsymbol{I}_p,n_1\right)$ and $\boldsymbol{M}_2\sim W_p\left(\boldsymbol{I}_p,n_2\right)$ be independently distributed. Let $\boldsymbol{M}_1^*$ be a lower triangular factorization of $\boldsymbol{M}_1$ such that $\boldsymbol{M}_1^*\boldsymbol{M}_1^{*\prime}=\boldsymbol{M}_1$. Let $\boldsymbol{Y}=\left(\boldsymbol{M}_1^*\right)^{-1}\boldsymbol{M}_2\left(\boldsymbol{M}_1^{*\prime}\right)^{-1}$. Then $\boldsymbol{Y}$ and $\boldsymbol{M}_1+\boldsymbol{M}_2$ are independent and the distribution of $\boldsymbol{Y}$ is*

$$\frac{\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_p\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\boldsymbol{Y}\right|^{(n_1-p-1)/2}\left|\boldsymbol{I}_p+\boldsymbol{Y}\right|^{-(n_1+n_2-p-1)/2}\frac{1}{\prod_{i=1}^{p}\left|\left(\boldsymbol{I}_p+\boldsymbol{Y}\right)^{[i]}\right|}$$

where $\left(\boldsymbol{I}_p+\boldsymbol{Y}\right)^{[i]}$ denotes the $i\times i$ upper left submatrix of $\left(\boldsymbol{I}_p+\boldsymbol{Y}\right)$.

Theorem 4.43. *Let $\boldsymbol{M}_1\sim W_p\left(\boldsymbol{I}_p,n_1\right)$ and $\boldsymbol{M}_2\sim W_p\left(\boldsymbol{I}_p,n_2\right)$ be independently distributed. Let $\boldsymbol{Y}=\boldsymbol{M}_1^{-1/2}\boldsymbol{M}_2\boldsymbol{M}_1^{-1/2}$. Then $\boldsymbol{Y}$ and $\boldsymbol{M}=\boldsymbol{M}_1+\boldsymbol{M}_2$ are not independent and their joint distribution is given by*

$$\begin{aligned}
&\frac{1}{2^{(n_1+n_2)p/2}\Gamma_p\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\boldsymbol{Y}\right|^{(n_1-p-1)/2}\left|\boldsymbol{I}_p+\boldsymbol{Y}\right|^{-(n_1+n_2-p-1)/2}\left|\boldsymbol{M}\right|^{(n_1+n_2-p-1)/2} \\
&\qquad\cdot\exp\left(-\frac{1}{2}\operatorname{Tr}\boldsymbol{M}\right)\prod_{i\le j}^{p}\frac{\left(\theta_i+\theta_j\right)}{\left(\eta_i+\eta_j\right)},
\end{aligned}\tag{4.42}$$

where $\theta_1 \geq \theta_2 \geq \ldots \geq \theta_p$ *are the ordered eigenvalues of* $\boldsymbol{M}_1^{1/2}$ *and* $\eta_1 \geq \eta_2 \geq \ldots \geq \eta_p$ *are the ordered eigenvalues of* $\left(\boldsymbol{I}_p + \boldsymbol{Y}\right)^{1/2} \boldsymbol{M}_1^{1/2} \left(\boldsymbol{I}_p + \boldsymbol{Y}\right)^{1/2}$.

Similar to the pseudo-Wishart distribution, a singular multivariate beta type II distribution may be defined for random matrices of less than full rank.

Definition 4.44 (**Singular Multivariate Beta Distribution, Type II**). *The* $p \times p$ *positive semidefinite, symmetric random matrix* $\boldsymbol{M}$ *of rank* n_1 *has a singular multivariate Beta type II distribution with* $n_1 < p$ *and* $n_2 \geq p$ *degrees of freedom, denoted as* $\boldsymbol{M} \sim M\beta_{II}^{n_1}(p, n_1, n_2)$*, if the probability density of* $\boldsymbol{M}$ *is*

$$\frac{\pi^{n_1(n_1-p)/2}\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_{n_1}\left(n_1/2\right)\Gamma_p\left(n_2/2\right)} \left|\boldsymbol{L}\right|^{(n_1-p-1)/2} \left|\boldsymbol{I}_p + \boldsymbol{M}\right|^{-(n_1+n_2)/2}$$

where $\boldsymbol{L} = \operatorname{diag}\left(\lambda_1, \lambda_2, \ldots, \lambda_{n_1}\right)$ *with* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n_1}$ *being the* n_1 *nonzero eigenvalues of* $\boldsymbol{M}$.

The derivation of the singular multivariate Beta type II distribution is given by Srivastava (2003). It extends the multivariate beta type II distribution to cases where the number of observations is less than the number of variables. If the singular matrix $\boldsymbol{M}_1 \sim W_p\left(\boldsymbol{I}_p, n_1\right)$ and the nonsingular matrix $\boldsymbol{M}_2 \sim W_p\left(\boldsymbol{I}_p, n_2\right)$, then $\boldsymbol{M}_2^{-1/2}\boldsymbol{M}_1\boldsymbol{M}_2^{-1/2\prime} \sim M\beta_{II}^{n_1}(p, n_1, n_2)$.

As with the nonsingular multivariate beta type II distribution, the singular multivariate beta type II distribution may be generalized so that the matrix parameters of the Wishart distribution is not the identity matrix. Although this generalization is straightforward, the following two definitions appear to be new to the literature.

Definition 4.45 (**Generalized Singular Multivariate Beta Distribution, Type II**). *The* $p \times p$ *positive semidefinite, symmetric random matrix* $\boldsymbol{M}$ *of rank* n_1 *has a generalized singular multivariate Beta type II distribution with* $n_1 < p$ *and* $n_2 \geq p$ *degrees of freedom and scale parameter* $\boldsymbol{\Omega}$*, denoted as* $\boldsymbol{M} \sim G\beta_{II}^{n_1}(p, n_1, n_2; \Omega)$*, if the probability density of* $\boldsymbol{M}$ *is*

$$\frac{\pi^{n_1(n_1-p)/2}\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_{n_1}\left(n_1/2\right)\Gamma_p\left(n_2/2\right)} \left|\boldsymbol{\Omega}\right|^{n_2/2} \left|\boldsymbol{L}\right|^{(n_1-p-1)/2} \left|\boldsymbol{\Omega} + \boldsymbol{M}\right|^{-(n_1+n_2)/2}$$

where $\boldsymbol{L} = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{n_1})$ *with* $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n_1}$ *being the* n_1 *nonzero eigenvalues of* $\boldsymbol{M}$.

The generalized singular multivariate distribution can be obtained using Definition 4.44 and Lemma 4.32. Let $\boldsymbol{M}_1 \sim M\beta_{II}^{n_1}(p, n_1, n_2)$ and $\boldsymbol{M} = \boldsymbol{\Omega}^{1/2}\boldsymbol{M}_1\boldsymbol{\Omega}^{1/2}$. The density of $\boldsymbol{M}_1$ is given by

$$f(\boldsymbol{M}_1) = \frac{\pi^{n_1(n_1-p)/2}\Gamma_p((n_1+n_2)/2)}{\Gamma_{n_1}(n_1/2)\Gamma_p(n_2/2)} |\boldsymbol{L}_1|^{(n_1-p-1)/2} \left|\boldsymbol{I}_p + \boldsymbol{M}_1\right|^{-(n_1+n_2)/2},$$

where $\boldsymbol{L}_1$ is the diagonal matrix of eigenvalues of $\boldsymbol{M}_1$. Since $\boldsymbol{\Omega}$ is full rank, $\boldsymbol{M}_1 = \boldsymbol{\Omega}^{-1/2}\boldsymbol{M}\boldsymbol{\Omega}^{-1/2}$; and by Lemma 4.32, the Jacobian of the transformation is $|\boldsymbol{L}_1|^{p+1-n_1} |\boldsymbol{L}|^{n_1-p-1} |\boldsymbol{\Omega}|^{-n_1/2}$. With the change in variables, the density of $\boldsymbol{M}$ is given by

$$\begin{aligned}
f(\boldsymbol{M}) &= \frac{\pi^{n_1(n_1-p)/2}\Gamma_p((n_1+n_2)/2)}{\Gamma_{n_1}(n_1/2)\Gamma_p(n_2/2)} |\boldsymbol{L}_1|^{(n_1-p-1)/2} \left|\boldsymbol{I}_p + \boldsymbol{\Omega}^{-1/2}\boldsymbol{M}\boldsymbol{\Omega}^{-1/2}\right|^{-(n_1+n_2)/2} \\
&\quad \cdot |\boldsymbol{L}_1|^{p+1-n_1} |\boldsymbol{L}|^{n_1-p-1} |\boldsymbol{\Omega}|^{-n_1/2} \\
&= \frac{\pi^{n_1(n_1-p)/2}\Gamma_p((n_1+n_2)/2)}{\Gamma_{n_1}(n_1/2)\Gamma_p(n_2/2)} |\boldsymbol{\Omega}|^{-n_1/2} |\boldsymbol{L}|^{(n_1-p-1)/2} \left|\boldsymbol{I}_p + \boldsymbol{\Omega}^{-1/2}\boldsymbol{M}\boldsymbol{\Omega}^{-1/2}\right|^{-(n_1+n_2)/2} \\
&= \frac{\pi^{n_1(n_1-p)/2}\Gamma_p((n_1+n_2)/2)}{\Gamma_{n_1}(n_1/2)\Gamma_p(n_2/2)} |\boldsymbol{\Omega}|^{-n_1/2} |\boldsymbol{L}|^{(n_1-p-1)/2} \left|\boldsymbol{I}_p + \boldsymbol{\Omega}^{-1/2}\boldsymbol{M}\boldsymbol{\Omega}^{-1/2}\right|^{-(n_1+n_2)/2} \\
&\quad \cdot |\boldsymbol{\Omega}|^{(n_1+n_2)/2} |\boldsymbol{\Omega}|^{-(n_1+n_2)/2} \\
&= \frac{\pi^{n_1(n_1-p)/2}\Gamma_p((n_1+n_2)/2)}{\Gamma_{n_1}(n_1/2)\Gamma_p(n_2/2)} |\boldsymbol{\Omega}|^{-n_1/2} |\boldsymbol{\Omega}|^{(n_1+n_2)/2} |\boldsymbol{L}|^{(n_1-p-1)/2} \\
&\quad \cdot \left|\boldsymbol{\Omega}^{1/2}\right|^{-(n_1+n_2)/2} \left|\boldsymbol{I}_p + \boldsymbol{\Omega}^{-1/2}\boldsymbol{M}\boldsymbol{\Omega}^{-1/2}\right|^{-(n_1+n_2)/2} \left|\boldsymbol{\Omega}^{1/2}\right|^{-(n_1+n_2)/2} \\
&= \frac{\pi^{n_1(n_1-p)/2}\Gamma_p((n_1+n_2)/2)}{\Gamma_{n_1}(n_1/2)\Gamma_p(n_2/2)} |\boldsymbol{\Omega}|^{n_2/2} |\boldsymbol{L}|^{(n_1-p-1)/2} |\boldsymbol{\Omega} + \boldsymbol{M}|^{-(n_1+n_2)/2},
\end{aligned} \tag{4.43}$$

which establishes the definition.

The singular multivariate F distribution may be derived from the generalized singular multivariate beta type II distribution in a manner analogous to their nonsingular counterparts.

Definition 4.46 (**Singular Multivariate F Distribution**). *The* $p \times p$ *positive semidefinite, symmetric random matrix* $\boldsymbol{M}$ *of rank* n_1 *has a singular multivariate F distribution with* $n_1 < p$ *and* $n_2 \geq p$ *degrees of freedom and scale matrix* $\boldsymbol{\Omega}$*, denoted as* $\boldsymbol{M} \sim F_p^{n_1}(n_1, n_2, \Omega)$*, if the probability*

density of $\boldsymbol{M}$ *is*

$$\frac{\pi^{n_1(n_1-p)/2}\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_{n_1}\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\Omega\right|^{-n_1/2}\left|\boldsymbol{L}\right|^{(n_1-p-1)/2}\left|\boldsymbol{I}_p+\boldsymbol{\Omega}^{-1}\boldsymbol{M}\right|^{-(n_1+n_2)/2}$$

where $\boldsymbol{L}=\operatorname{diag}\left(\lambda_1,\lambda_2,\ldots,\lambda_{n_1}\right)$ *with* $\lambda_1\geq\lambda_2\geq\cdots\geq\lambda_{n_1}$ *being the* n_1 *nonzero eigenvalues of* $\boldsymbol{M}$.

This definition can be obtained directly from Equation (4.43):

$$\begin{aligned}
f\left(\boldsymbol{M}\right) &= \frac{\pi^{n_1(n_1-p)/2}\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_{n_1}\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\boldsymbol{\Omega}\right|^{-n_1/2}\left|\boldsymbol{L}\right|^{(n_1-p-1)/2}\left|\boldsymbol{I}_p+\boldsymbol{\Omega}^{-1/2}\boldsymbol{M}\boldsymbol{\Omega}^{-1/2}\right|^{-(n_1+n_2)/2} \\
&= \frac{\pi^{n_1(n_1-p)/2}\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_{n_1}\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\boldsymbol{\Omega}\right|^{-n_1/2}\left|\boldsymbol{L}\right|^{(n_1-p-1)/2}\left|\boldsymbol{I}_p+\boldsymbol{\Omega}^{-1/2}\boldsymbol{M}\boldsymbol{\Omega}^{-1/2}\right|^{-(n_1+n_2)/2} \\
&\quad\cdot\left|\boldsymbol{\Omega}^{-1/2}\right|^{-(n_1+n_2)/2}\left|\boldsymbol{\Omega}^{1/2}\right|^{-(n_1+n_2)/2} \\
&= \frac{\pi^{n_1(n_1-p)/2}\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_{n_1}\left(n_1/2\right)\Gamma_p\left(n_2/2\right)}\left|\boldsymbol{\Omega}\right|^{-n_1/2}\left|\boldsymbol{L}\right|^{(n_1-p-1)/2}\left|\boldsymbol{\Omega}^{-1/2}\right|^{-(n_1+n_2)/2} \\
&\quad\cdot\left|\boldsymbol{I}_p+\boldsymbol{\Omega}^{-1/2}\boldsymbol{M}\boldsymbol{\Omega}^{-1/2}\right|^{-(n_1+n_2)/2}\left|\boldsymbol{\Omega}^{1/2}\right|^{-(n_1+n_2)/2} \\
&= \frac{\pi^{n_1(n_1-p)/2}\Gamma_p\left(\left(n_1+n_2\right)/2\right)}{\Gamma_{n_1}\left(n_1/2\right)\Gamma_p\left(-n_1/2\right)}\left|\boldsymbol{L}\right|^{(n_1-p-1)/2}\left|\boldsymbol{I}_p+\boldsymbol{\Omega}^{-1}\boldsymbol{M}\right|^{-(n_1+n_2)/2},
\end{aligned}$$

which establishes the definition.

Definition 4.47 (**Wilks' Lambda Distribution**). *Let* $\boldsymbol{M}_1\sim W_p\left(\boldsymbol{I}_p,m\right)$ *be a random matrix and* $\boldsymbol{M}_2\sim W_p\left(\boldsymbol{I}_p,n\right)$ *be a random matrix independent of* $\boldsymbol{M}_1$. *Then the test statistic*

$$\Lambda=\frac{\left|\boldsymbol{M}_1\right|}{\left|\boldsymbol{M}_1+\boldsymbol{M}_2\right|}=\left|\boldsymbol{I}_p+\boldsymbol{M}_1^{-1}\boldsymbol{M}_2\right|^{-1}$$

has a Wilks' lambda distribution with m and n degrees of freedom, denoted as $\Lambda\left(p,m,n\right)$.

The Wilks' lambda statistic is a multivariate generalization of the F statistic for conducting likelihood ratio tests about the values of the random vectors from which the covariance matrices $\boldsymbol{M}_1$ and $\boldsymbol{M}_2$ are derived. However, Wilks' lambda rejects for small values of the test statistic, while the F test rejects for large values.

4.3.4 Development of the Multivariate Random Variance Model

In generalizing the univariate RVM to a multivariate one, a general linear mixed model will be used, with the univariate variables replaced by multivariate counterparts. For the usual multivariate model, the intensities Y will be modeled as

$$\underset{n\times p}{\boldsymbol{Y}} = \underset{n\times k}{\boldsymbol{X}'}\,\underset{k\times p}{\boldsymbol{\beta}} + \underset{n\times p}{\boldsymbol{\varepsilon}} \tag{4.44}$$

where $\boldsymbol{Y}$ now represents a matrix of probe-level intensities within a probeset for each chip rather than a vector of probeset-level intensities for each probeset on a chip. $\boldsymbol{X}$ remains the design matrix for the experiment, $\boldsymbol{\beta}$ the coefficients representing the effects of probes, and $\boldsymbol{\varepsilon}$ random error. It is assumed that $\boldsymbol{\varepsilon}$ has a p-variate normal distribution with mean $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$, which will be denoted as $\boldsymbol{\varepsilon} \sim N_p(\mathbf{0}, \boldsymbol{\Sigma})$. As in the univariate case, the usual multivariate model has the assumption that $\boldsymbol{\Sigma}$ is a constant term. In the multivariate RVM, the covariance matrix is allowed to vary for each probeset, and the set of covariance matrices are modeled as realizations from an underlying distribution. The Wishart distribution was chosen as the prior distribution for the inverse of the covariance matrices, analogous to the gamma distribution in the univariate RVM, so that

$$\boldsymbol{\Sigma}^{-1} \sim W_p\left(\boldsymbol{\Sigma}^{-1}; \boldsymbol{B}, a\right) \equiv \frac{\left|\boldsymbol{\Sigma}^{-1}\right|^{(a-p-1)/2}}{2^{ap/2}\,|\boldsymbol{B}|^{a/2}\,\Gamma_p\left(\frac{a}{2}\right)} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{B}^{-1}\boldsymbol{\Sigma}^{-1}\right]\right). \tag{4.45}$$

The scalar hyperparameter a, which represents the degrees of freedom, and the matrix hyperparameter $\boldsymbol{B}$, which represents the covariance of the Wishart distribution, are estimated from the sample data. The assumption in Equation (4.45) leads to a modification of the sums of squares and the test statistics similar to that of the univariate RVM. With the model in Equation (4.44), the hypotheses of interest are $H_0 : \boldsymbol{\beta} \in \omega$ versus $H_1 : \boldsymbol{\beta} \in \mathbb{R}^{p\times p}$, where ω is a linear subspace of $\mathbb{R}^{p\times p}$. In the usual multivariate linear models, when there are no distributional assumptions on the

covariance matrix $\boldsymbol{\Sigma}$, the maximum likelihood estimate for $\boldsymbol{\beta}$ under H_1 is

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}, \tag{4.46}$$

where $\boldsymbol{X}$ is of rank k. Under H_0 the MLE is

$$\widehat{\widehat{\boldsymbol{\beta}}} = (\boldsymbol{X}'_\omega\boldsymbol{X}_\omega)^{-1}\boldsymbol{X}'_\omega\boldsymbol{Y} \tag{4.47}$$

where $\boldsymbol{X}_\omega$, a matrix of rank r, is again the design matrix $\boldsymbol{X}$ projected into the subspace ω. The multivariate RVM test is given by the following theorem, which is an extension of Theorem 3.1.

Theorem 4.48. *Under multivariate RVM, the likelihood ratio test statistic for testing $H_0 : \boldsymbol{\beta} \in \omega$ against $H_1 : \boldsymbol{\beta} \in \mathbb{R}^{p\times p}$ will be of the form,*

$$\widetilde{\Lambda}^{(n+a-p-1)/2} = \frac{\left|\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right|}{\left|\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right|} > \delta^{(n+a-p-1)/2} \tag{4.48}$$

where $\widehat{\widehat{\boldsymbol{SS}}}$ and $\widehat{\boldsymbol{SS}}$ are the sums of squares and crossproducts under H_0 and H_1, respectively, defined as

$$\widehat{\boldsymbol{SS}} = \left(\boldsymbol{Y} - \boldsymbol{X}'\widehat{\boldsymbol{\beta}}\right)'\left(\boldsymbol{Y} - \boldsymbol{X}'\widehat{\boldsymbol{\beta}}\right), \tag{4.49}$$

$$\widehat{\widehat{\boldsymbol{SS}}} = \left(\boldsymbol{Y} - \boldsymbol{X}'_\omega\widehat{\widehat{\boldsymbol{\beta}}}\right)'\left(\boldsymbol{Y} - \boldsymbol{X}'_\omega\widehat{\widehat{\boldsymbol{\beta}}}\right), \tag{4.50}$$

and $\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}$ is the sums of squares and crossproducts under RVM.

Proof. Let $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n$ and $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ denote vectors corresponding to the rows of $\boldsymbol{Y}$ and $\boldsymbol{X}$, respectively, so that $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n]'$ and $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n]'$. Then $\boldsymbol{y}_j \sim N_p\left(\boldsymbol{\beta}'\boldsymbol{x}_j, \boldsymbol{\Sigma}\right)$ for

$j = 1, 2, \ldots, n$ and are independent of each other. The density function of $\boldsymbol{Y}$ is

$$L(\boldsymbol{Y}|\boldsymbol{\beta}, \boldsymbol{\Sigma}) = \prod_{j=1}^{n} \frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)' \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)\right)$$
$$= \frac{1}{(2\pi)^{np/2} |\boldsymbol{\Sigma}|^{n/2}} \exp\left(-\frac{1}{2}\sum_{j=1}^{n}\left[\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)' \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)\right]\right). \quad (4.51)$$

Since a scalar is equal to its trace, $\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)' \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right) = \mathrm{Tr}\left[\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)' \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)\right]$. Using Property (4) of the trace, it follows that

$$\sum_{j=1}^{n}\left[\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)' \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)\right] = \sum_{j=1}^{n}\mathrm{Tr}\left[\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)' \boldsymbol{\Sigma}^{-1}\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)\right]$$
$$= \sum_{j=1}^{n}\mathrm{Tr}\left[\boldsymbol{\Sigma}^{-1}\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)\left(\boldsymbol{y}_j - \boldsymbol{\beta}'\boldsymbol{x}_j\right)'\right]$$
$$= \mathrm{Tr}\left[\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})\right]$$

since

$$\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta} = [\boldsymbol{y}_1 - \boldsymbol{\beta}'\boldsymbol{x}_1, \boldsymbol{y}_2 - \boldsymbol{\beta}'\boldsymbol{x}_2, \ldots, \boldsymbol{y}_n - \boldsymbol{\beta}'\boldsymbol{x}_n]'.$$

Thus the density function may be rewritten as

$$L(\boldsymbol{Y}|\boldsymbol{\beta}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{np/2} |\boldsymbol{\Sigma}|^{n/2}} \exp\left(-\frac{1}{2}\mathrm{Tr}\left[\boldsymbol{\Sigma}^{-1}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})\right]\right). \quad (4.52)$$

Let $\boldsymbol{Z} = \boldsymbol{\Sigma}^{-1}$. Include the distributional assumptions as presented in Equation (4.45) and the density in Equation (4.52) becomes

$$L(\boldsymbol{Y}|\boldsymbol{\beta}, \boldsymbol{Z}) = \frac{|\boldsymbol{Z}|^{n/2}}{(2\pi)^{np/2}} \exp\left(-\frac{1}{2}\mathrm{Tr}\left[\boldsymbol{Z}(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y} - \boldsymbol{X}\boldsymbol{\beta})\right]\right) \cdot W_p(\boldsymbol{Z}; \boldsymbol{B}, a)$$

This is equivalent to

$$L(\boldsymbol{Y}|\boldsymbol{\beta},\boldsymbol{Z}) = \frac{|\boldsymbol{Z}|^{n/2}}{(2\pi)^{np/2}} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{Z}\left(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right)'\left(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right)\right]\right) \cdot \frac{|\boldsymbol{Z}|^{(a-p-1)/2}}{2^{ap/2}\,|\boldsymbol{B}|^{a/2}\,\Gamma_p\left(\frac{a}{2}\right)} \exp\left(-\frac{1}{2}\operatorname{Tr}\boldsymbol{B}^{-1}\boldsymbol{Z}\right) \tag{4.53}$$

after explicitly stating the Wishart distribution. This simplifies to

$$L(\boldsymbol{Y}|\boldsymbol{\beta},\boldsymbol{Z}) = \frac{|\boldsymbol{Z}|^{(n+a-p-1)/2}}{2^{(n+a)p/2}\pi^{np/2}\,|\boldsymbol{B}|^{a/2}\,\Gamma_p\left(\frac{a}{2}\right)} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{Z}\left(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right)'\left(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right)\right] - \frac{1}{2}\operatorname{Tr}\boldsymbol{B}^{-1}\boldsymbol{Z}\right)$$

or

$$L(\boldsymbol{Y}|\boldsymbol{\beta},\boldsymbol{Z}) = c_1\,|\boldsymbol{Z}|^{(n+a-p-1)/2} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{Z}\left(\left(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right)'\left(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right)+\boldsymbol{B}^{-1}\right)\right]\right)$$

where the term

$$c_1 = \frac{1}{2^{(n+a)p/2}\pi^{np/2}\,|\boldsymbol{B}|^{a/2}\,\Gamma_p\left(\frac{a}{2}\right)}$$

is a scaling constant. The joint likelihood of $\boldsymbol{\beta}$ and $\boldsymbol{Z}$ is

$$L(\boldsymbol{\beta},\boldsymbol{Z}|\boldsymbol{Y}) = c_1\,|\boldsymbol{Z}|^{(n+a-p-1)/2} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{Z}\left(\left(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right)'\left(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right)+\boldsymbol{B}^{-1}\right)\right]\right)$$

with log likelihood

$$\begin{aligned}\log L(\boldsymbol{\beta},\boldsymbol{Z}|\boldsymbol{Y}) &= \log(c_1) + \frac{n+a-p-1}{2}\log|\boldsymbol{Z}| - \frac{1}{2}\operatorname{Tr}\left(\boldsymbol{Z}\left[\left(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right)'\left(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right)+\boldsymbol{B}^{-1}\right]\right)\\ &= \log(c_1) + \frac{n+a-p-1}{2}\log|\boldsymbol{Z}| - \frac{1}{2}\operatorname{Tr}\left[\boldsymbol{Z}\left(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right)'\left(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}\right)\right] - \frac{1}{2}\operatorname{Tr}\left(\boldsymbol{Z}\boldsymbol{B}^{-1}\right)\end{aligned}$$

Using Lemma 4.24 and noting that $\boldsymbol{Z}$ is symmetric, the differential with respect to $\boldsymbol{\beta}$ is

$$\begin{aligned}
d \log L(\boldsymbol{\beta}, \boldsymbol{Z}|\boldsymbol{Y}) &= d\left(-\frac{1}{2} \operatorname{Tr}\left[\boldsymbol{Z}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})\right]\right) \\
&= -\frac{1}{2} \operatorname{Tr}\left[d\boldsymbol{Z}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})\right] \\
&= -\frac{1}{2} \operatorname{Tr}\left[(\boldsymbol{Z}+\boldsymbol{Z}')(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})' d(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})\right] \\
&= -\frac{1}{2} \operatorname{Tr}\left[(\boldsymbol{Z}+\boldsymbol{Z}')(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'(-\boldsymbol{X})\, d\boldsymbol{\beta}\right] \\
&= \frac{1}{2} \operatorname{Tr}\left[2\boldsymbol{Z}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'\boldsymbol{X} d\boldsymbol{\beta}\right] \\
&= \operatorname{Tr}\left[\boldsymbol{Z}(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'\boldsymbol{X} d\boldsymbol{\beta}\right] \\
&= \operatorname{Tr}\left[(\boldsymbol{X}'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})\boldsymbol{Z}')' d\boldsymbol{\beta}\right] \\
&= \operatorname{Tr}\left[(\boldsymbol{X}'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})\boldsymbol{Z})' d\boldsymbol{\beta}\right]
\end{aligned}$$

Using Equation (4.9), this becomes

$$\operatorname{Tr}\left[(\boldsymbol{X}'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})\boldsymbol{Z})' d\boldsymbol{\beta}\right] = (\operatorname{vec}\left[\boldsymbol{X}'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})\boldsymbol{Z}\right])' \operatorname{vec}(d\boldsymbol{\beta})$$

so that, by the First Identification Theorem,

$$\frac{\partial}{\partial (\operatorname{vec}(\boldsymbol{\beta}))'} \log L(\boldsymbol{\beta}, \boldsymbol{Z}|\boldsymbol{Y}) = (\operatorname{vec}\left[\boldsymbol{X}'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})\boldsymbol{Z}\right])' \tag{4.54}$$

To obtain the maximum likelihood estimate $\widehat{\widehat{\boldsymbol{\beta}}}$ under H_0, setting Equation (4.54) to $\mathbf{0}$ and noting $\boldsymbol{Z} \neq \mathbf{0}$ implies

$$\boldsymbol{X}'\left(\boldsymbol{Y}-\boldsymbol{X}\widehat{\widehat{\boldsymbol{\beta}}}\right) = \mathbf{0}$$

Thus

$$\widehat{\widehat{\boldsymbol{\beta}}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$$

The differential with respect to $\boldsymbol{Z}$ is

$$\begin{aligned} d \log L(\boldsymbol{\beta}, \boldsymbol{Z}|\boldsymbol{Y}) &= d\left[\frac{n+a-p-1}{2} \log |\boldsymbol{Z}| - \frac{1}{2} \operatorname{Tr}\left(\boldsymbol{Z}\left[(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}) + \boldsymbol{B}^{-1}\right]\right)\right] \\ &= d\left(\frac{n+a-p-1}{2} \log |\boldsymbol{Z}|\right) - d\left[\frac{1}{2} \operatorname{Tr}\left(\boldsymbol{Z}\left[(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}) + \boldsymbol{B}^{-1}\right]\right)\right] \end{aligned} \tag{4.55}$$

To obtain the maximum likelihood estimate $\widehat{\widehat{\boldsymbol{Z}}}$, note that

$$\begin{aligned} d \log |\boldsymbol{Z}| &= \operatorname{Tr}\left(\boldsymbol{Z}^{-1}\right) d\boldsymbol{Z} \\ &= \left[\boldsymbol{D}_p' \operatorname{vec}\left(\boldsymbol{Z}^{-1}\right)\right]' \operatorname{vech}(d\boldsymbol{Z}) \end{aligned} \tag{4.56}$$

using Lemma 4.25 and

$$\begin{aligned} d \operatorname{Tr}\left(\boldsymbol{Z}\left[(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}) + \boldsymbol{B}^{-1}\right]\right) &= \operatorname{Tr}\left(\left[(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}) + \boldsymbol{B}^{-1}\right] d\boldsymbol{Z}\right) \\ &= \left(\boldsymbol{D}_p' \operatorname{vec}\left[(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}) + \boldsymbol{B}^{-1}\right]\right)' \operatorname{vech}(d\boldsymbol{Z}) \end{aligned} \tag{4.57}$$

using Lemma 4.24. Substituting Equations (4.56) and (4.57) into Equation (4.55) yields

$$\begin{aligned} d \log L(\boldsymbol{\beta}, \boldsymbol{Z}|\boldsymbol{Y}) = &\frac{n+a-p-1}{2}\left[\boldsymbol{D}_p' \operatorname{vec}\left(\boldsymbol{Z}^{-1}\right)\right]' \operatorname{vech}(d\boldsymbol{Z}) \\ &- \frac{1}{2}\left(\boldsymbol{D}_p' \operatorname{vec}\left[(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}) + \boldsymbol{B}^{-1}\right]\right)' \operatorname{vech}(d\boldsymbol{Z}) \end{aligned}$$

so that

$$\begin{aligned} \frac{\partial}{\partial [\operatorname{vech}(\boldsymbol{Z})]'} \log L(\boldsymbol{\beta}, \boldsymbol{Z}|\boldsymbol{Y}) = &\frac{n+a-p-1}{2}\left[\boldsymbol{D}_p' \operatorname{vec}\left(\boldsymbol{Z}^{-1}\right)\right]' \\ &- \frac{1}{2}\left(\boldsymbol{D}_p' \operatorname{vec}\left[(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta})'(\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{\beta}) + \boldsymbol{B}^{-1}\right]\right)' \end{aligned}$$

by the First Identification Theorem. Substituting $\widehat{\widehat{\boldsymbol{S}\boldsymbol{S}}}$ from Equation (4.50) gives

$$\frac{\partial}{\partial [\operatorname{vech}(\boldsymbol{Z})]'} \log L(\boldsymbol{\beta}, \boldsymbol{Z}|\boldsymbol{Y}) = \frac{n+a-p-1}{2}\left[\boldsymbol{D}_p' \operatorname{vec}\left(\boldsymbol{Z}^{-1}\right)\right]' - \frac{1}{2}\left[\boldsymbol{D}_p' \operatorname{vec}\left(\widehat{\widehat{\boldsymbol{S}\boldsymbol{S}}} + \boldsymbol{B}^{-1}\right)\right]'$$

This can be rearranged to give

$$\begin{aligned}\frac{\partial}{\partial\left[\operatorname{vec}(\boldsymbol{Z})\right]'}\log L(\boldsymbol{\beta},\boldsymbol{Z}|\boldsymbol{Y}) &= \left[\boldsymbol{D}_p'\operatorname{vec}\left(\frac{n+a-p-1}{2}\boldsymbol{Z}^{-1}\right)\right]' - \left(\boldsymbol{D}_p'\operatorname{vec}\left[\frac{1}{2}\left(\widehat{\widehat{\boldsymbol{S}\boldsymbol{S}}}+\boldsymbol{B}^{-1}\right)\right]\right)' \\ &= \left(\boldsymbol{D}_p'\operatorname{vec}\left(\frac{n+a-p-1}{2}\boldsymbol{Z}^{-1}\right) - \boldsymbol{D}_p'\operatorname{vec}\left[\frac{1}{2}\left(\widehat{\widehat{\boldsymbol{S}\boldsymbol{S}}}+\boldsymbol{B}^{-1}\right)\right]\right)' \\ &= \left(\boldsymbol{D}_p'\operatorname{vec}\left[\frac{n+a-p-1}{2}\boldsymbol{Z}^{-1} - \frac{1}{2}\left(\widehat{\widehat{\boldsymbol{S}\boldsymbol{S}}}+\boldsymbol{B}^{-1}\right)\right]\right)'\end{aligned}$$

Let

$$\boldsymbol{O} = \frac{n+a-p-1}{2}\boldsymbol{Z}^{-1} - \frac{1}{2}\left(\widehat{\widehat{\boldsymbol{S}\boldsymbol{S}}}+\boldsymbol{B}^{-1}\right)$$

so that

$$\begin{aligned}\frac{\partial}{\partial\left[\operatorname{vech}(\boldsymbol{Z})\right]'}\log L(\boldsymbol{\beta},\boldsymbol{Z}|\boldsymbol{Y}) &= \left[\boldsymbol{D}_p'\operatorname{vec}(\boldsymbol{O})\right]' \\ &= (\operatorname{vech}\left[\boldsymbol{O}+\boldsymbol{O}'-\operatorname{diag}(\boldsymbol{O})\right])' \qquad (4.58)\end{aligned}$$

Setting this to $\mathbf{0}$ and noting that $\boldsymbol{O}$ is symmetric implies

$$2\boldsymbol{O} - \operatorname{diag}(\boldsymbol{O}) = \mathbf{0}$$

which in turn implies $\boldsymbol{O} = \mathbf{0}$. (For the diagonal elements o_{ii} of $\boldsymbol{O}$, $2o_{ii} - o_{ii} = 0 \Rightarrow o_{ii} = 0$, while for the off-diagonal elements $2o_{ij} - 0 = 0 \Rightarrow o_{ij} = 0$.) Thus

$$\frac{n+a-p-1}{2}\widehat{\widehat{\boldsymbol{Z}}}^{-1} - \frac{1}{2}\left(\widehat{\widehat{\boldsymbol{S}\boldsymbol{S}}}+\boldsymbol{B}^{-1}\right) = \mathbf{0}$$

Solving for $\widehat{\widehat{\boldsymbol{Z}}}^{-1}$ gives

$$\widehat{\widehat{\boldsymbol{Z}}}^{-1} = \frac{1}{n+a-p-1}\left(\widehat{\widehat{\boldsymbol{S}\boldsymbol{S}}}+\boldsymbol{B}^{-1}\right)$$

or

$$\widehat{\widehat{\boldsymbol{Z}}} = (n + a - p - 1)\left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right)^{-1}$$

The corresponding maximum likelihood is

$$\max_{H \in H_0} L(\boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{\beta}, a, \boldsymbol{B}) = \frac{\left|(n + a - p - 1)\left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right)^{-1}\right|^{(n+a-p-1)/2}}{(2\pi)^{(n+a)p/2}\,|\boldsymbol{B}|^{a/2}\,\Gamma_p\left(\frac{a}{2}\right)} \cdot \exp\left(-\frac{1}{2}\operatorname{Tr}\left[(n + a - p - 1)\left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right)^{-1}\left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right)\right]\right).$$

Noting that $\left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right)^{-1}\left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right) = \boldsymbol{I}_p$ and $\operatorname{Tr}[(n + a - p - 1)\,\boldsymbol{I}_p] = (n + a - p - 1)\operatorname{Tr}\boldsymbol{I}_p = (n + a - p - 1)\,p$, the likelihood can be rewritten as

$$\max_{H \in H_0} L(\boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{\beta}, a, \boldsymbol{B}) = c_2\left(\frac{1}{\left|\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right|}\right)^{(n+a-p-1)/2} \tag{4.59}$$

where

$$c_2 = \frac{(n + a - p - 1)^{(n+a-p-1)p/2}}{(2\pi)^{(n+a)p/2}\,|\boldsymbol{B}|^{a/2}\,\Gamma_p\left(\frac{a}{2}\right)} \exp\left(-\frac{(n + a - p - 1)\,p}{2}\right)$$

is a constant.

Similarly, under H_1, the maximum likelihood estimates are

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\,\boldsymbol{X}'\boldsymbol{Y}$$

and

$$\widehat{\boldsymbol{Z}} = \frac{1}{n + a - p - 1}\left(\widehat{\boldsymbol{SS}} - \boldsymbol{B}^{-1}\right)$$

with the maximum likelihood

$$\begin{aligned}\max_{H\in H_1} L(\boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{\beta}, a, \boldsymbol{B}) &= \frac{\left|(n+a-p-1)\left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right)^{-1}\right|^{(n+a-p-1)/2}}{(2\pi)^{(n+a)p/2}\,|\boldsymbol{B}|^{a/2}\,\Gamma_p\left(\frac{a}{2}\right)} \\ &\quad \cdot \exp\left(-\frac{1}{2}\operatorname{Tr}\left[(n+a-p-1)\left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right)^{-1}\left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right)\right]\right) \\ &= c_2 \left(\frac{1}{\left|\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right|}\right)^{(n+a-p-1)/2}. \end{aligned} \tag{4.60}$$

The likelihood ratio test is

$$\widetilde{\Lambda} = \frac{\max_{H\in H_0} L(\boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{\beta}, a, \boldsymbol{B})}{\max_{H\in H_1} L(\boldsymbol{Y} \mid \boldsymbol{Z}, \boldsymbol{\beta}, a, \boldsymbol{B})} > \delta \tag{4.61}$$

Substituting Equations (4.59) and (4.60) into Equation (4.61) yields

$$\widetilde{\Lambda} = \left(\frac{\left|\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right|}{\left|\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right|}\right)^{(n+a-p-1)/2} > \delta$$

which is equivalent to

$$\widetilde{\Lambda}^{(n+a-p-1)/2} = \frac{\left|\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right|}{\left|\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right|} > \delta^{(n+a-p-1)/2}.$$

Thus $\widetilde{\Lambda}$ is a likelihood ratio test. □

As with univariate RVM, additional information is needed before the likelihood ratio test in Equation (4.48) can be used in hypothesis testing. Define

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{n-k}\widehat{\boldsymbol{SS}}$$

and

$$\widehat{\widehat{\boldsymbol{\Sigma}}} = \frac{1}{n-r}\widehat{\widehat{\boldsymbol{SS}}}.$$

In GLM with no assumptions of $\boldsymbol{\Sigma}$, $\widehat{\widehat{\boldsymbol{\Sigma}}}$ is the residual (or "within") sums of squares under H_0, which is an unbiased estimator of $\boldsymbol{\Sigma}$. (As with the univariate RVM, the unbiased estimator of the variance-covariance matrix, rather than the maximum likelihood estimator, is used.) Furthermore, $(n-r)\,\widehat{\widehat{\boldsymbol{\Sigma}}} \sim W_p(\boldsymbol{\Sigma}, n-r)$. Similarly, $(n-k)\,\widehat{\boldsymbol{\Sigma}} \sim W_p(\boldsymbol{\Sigma}, n-k)$ is the residual sums of squares under H_1. Now define $\widetilde{\boldsymbol{SS}}$, which is the sums of squares and crossproducts under RVM, as

$$\widetilde{\boldsymbol{SS}} = \widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}.$$

Define

$$\widetilde{\boldsymbol{\Sigma}} = \frac{1}{n-k+a}\widetilde{\boldsymbol{SS}}.$$

and

$$\widetilde{\widetilde{\boldsymbol{\Sigma}}} = \frac{1}{n-r+a}\left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right)$$

as the mean sums of squares under RVM. The next theorem derives the distribution of $\widetilde{\boldsymbol{\Sigma}}$ (and, by an analogous argument, $\widetilde{\widetilde{\boldsymbol{\Sigma}}}$), which is used to establish the distribution of the likelihood ratio test statistic $\widetilde{\Lambda}$ and related test statistics. It also shows how estimates of a and $\boldsymbol{B}$ may be obtained from the sample data.

Theorem 4.49. *For $\widehat{\boldsymbol{\Sigma}}$ and $\widetilde{\boldsymbol{\Sigma}}$ as above,*

$$\boldsymbol{\Sigma}^{-1}\left((n-k+a)\,\widetilde{\boldsymbol{\Sigma}}\right) = \boldsymbol{\Sigma}^{-1}\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right) \sim W_p\left(\boldsymbol{I}_p, n-k+a\right)$$

and

$$(a+p-1)\,\boldsymbol{B}^{1/2}\widehat{\boldsymbol{\Sigma}}\boldsymbol{B}^{1/2} \sim F_p\left(n-k, a+p-1, \tfrac{a+p-1}{n-k}\boldsymbol{I}_p\right) \tag{4.62}$$
$$\equiv G\beta_{II}\left(p, n-k, a+p-1; \tfrac{n-k}{a+p-1}\boldsymbol{I}_p, \boldsymbol{0}\right)$$

Proof. From standard GLM theory, the density function of $(n-k)\,\widehat{\boldsymbol{\Sigma}}$ is

$$f\left((n-k)\,\widehat{\boldsymbol{\Sigma}}\right) = \frac{\left|(n-k)\,\widehat{\boldsymbol{\Sigma}}\right|^{(n-k-p-1)/2}\left|\boldsymbol{\Sigma}^{-1}\right|^{(n-k)/2}}{2^{(n-k)p/2}\Gamma_p\left(\frac{n-k}{2}\right)}\exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}(n-k)\,\widehat{\boldsymbol{\Sigma}}\right]\right)$$

so that

$$f\left(\widehat{\boldsymbol{\Sigma}}\right) = \frac{(n-k)\left|(n-k)\,\widehat{\boldsymbol{\Sigma}}\right|^{(n-k-p-1)/2}\left|\boldsymbol{\Sigma}^{-1}\right|^{(n-k)/2}}{2^{(n-k)p/2}\Gamma_p\left(\frac{n-k}{2}\right)}\exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}(n-k)\,\widehat{\boldsymbol{\Sigma}}\right]\right)$$

Analogous to univariate RVM, let $\boldsymbol{\Sigma}^{-1} \sim W_p(\boldsymbol{B}, a)$. The joint density of $\widehat{\boldsymbol{\Sigma}}$ and $\boldsymbol{\Sigma}^{-1}$ is

$$L\left(\widehat{\boldsymbol{\Sigma}}, \boldsymbol{\Sigma}^{-1}\right) = \frac{(n-k)\left|(n-k)\,\widehat{\boldsymbol{\Sigma}}\right|^{(n-k-p-1)/2}\left|\boldsymbol{\Sigma}^{-1}\right|^{(n-k)/2}}{2^{(n-k)p/2}\Gamma_p\left(\frac{n-k}{2}\right)}\exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}(n-k)\,\widehat{\boldsymbol{\Sigma}}\right]\right)W_p\left(\boldsymbol{\Sigma}^{-1}; \boldsymbol{B}, a\right),$$

which is

$$L\left(\widehat{\boldsymbol{\Sigma}}, \boldsymbol{\Sigma}^{-1}\right) = \frac{(n-k)\left|(n-k)\,\widehat{\boldsymbol{\Sigma}}\right|^{(n-k-p-1)/2}\left|\boldsymbol{\Sigma}^{-1}\right|^{(n-k)/2}}{2^{(n-k)p/2}\Gamma_p\left(\frac{n-k}{2}\right)}\exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}(n-k)\,\widehat{\boldsymbol{\Sigma}}\right]\right)$$
$$\cdot\frac{\left|\boldsymbol{\Sigma}^{-1}\right|^{(a-p-1)/2}}{2^{ap/2}\left|\boldsymbol{B}\right|^{a/2}\Gamma_p\left(\frac{a}{2}\right)}\exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{B}^{-1}\boldsymbol{\Sigma}^{-1}\right]\right)$$

after explicitly stating the Wishart distribution. This simplifies to

$$L\left(\widehat{\boldsymbol{\Sigma}}, \boldsymbol{\Sigma}^{-1}\right) = \frac{(n-k)^{(n-k-p+1)/2}}{2^{(n-k+a)p/2}\left|\boldsymbol{B}\right|^{a/2}\Gamma_p\left(\frac{n-k}{2}\right)\Gamma_p\left(\frac{a}{2}\right)}\left|\widehat{\boldsymbol{\Sigma}}\right|^{(n-k-p-1)/2}\left|\boldsymbol{\Sigma}^{-1}\right|^{(n-k+a-p-1)/2}$$
$$\cdot\exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}\left((n-k)\,\widehat{\boldsymbol{\Sigma}}+B^{-1}\right)\right]\right)$$

This is equivalent to

$$L\left(\widehat{\boldsymbol{\Sigma}}, \boldsymbol{\Sigma}^{-1}\right) = c_1 \frac{\left|\widehat{\boldsymbol{\Sigma}}\right|^{(n-k-p-1)/2}}{|\boldsymbol{\Sigma}|^{(n-k+a-p-1)/2}} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}\left((n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)\right]\right) \tag{4.63}$$

where

$$c_1 = \frac{(n-k)^{(n-k-p+1)/2}}{2^{(n-k+a)p/2}\,|\boldsymbol{B}|^{a/2}\,\Gamma_p\left(\frac{n-k}{2}\right)\Gamma_p\left(\frac{a}{2}\right)}$$

is a constant. Let

$$\boldsymbol{U} = \boldsymbol{\Sigma}^{-1}\left((n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right) \tag{4.64}$$

and

$$\boldsymbol{V} = (a+p-1)^{1/2}\,\boldsymbol{B}^{1/2}\widehat{\boldsymbol{\Sigma}}\boldsymbol{B}^{1/2}\,(a+p-1)^{1/2}\,. \tag{4.65}$$

This implies

$$\widehat{\boldsymbol{\Sigma}} = (a+p-1)^{-1/2}\,\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2}\,(a+p-1)^{-1/2}\,,$$

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{U}\left((n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)^{-1} = \boldsymbol{U}\left(\frac{n-k}{a+p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2} + \boldsymbol{B}^{-1}\right)^{-1}\,,$$

and

$$\boldsymbol{\Sigma} = \left(\frac{n-k}{a+p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2} + \boldsymbol{B}^{-1}\right)\boldsymbol{U}^{-1}.$$

The Jacobian of the transformation is

$$\begin{aligned} J_{\boldsymbol{\Sigma}^{-1},\widehat{\boldsymbol{\Sigma}}\to\boldsymbol{U},\boldsymbol{V}} &= \begin{vmatrix} \dfrac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\boldsymbol{U}'} & \dfrac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\boldsymbol{V}'} \\ \dfrac{\partial\widehat{\boldsymbol{\Sigma}}}{\partial\boldsymbol{U}'} & \dfrac{\partial\widehat{\boldsymbol{\Sigma}}}{\partial\boldsymbol{V}'} \end{vmatrix} \\ &= \begin{vmatrix} \dfrac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\boldsymbol{U}'} & \dfrac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\boldsymbol{V}'} \\ \mathbf{0} & \dfrac{\partial\widehat{\boldsymbol{\Sigma}}}{\partial\boldsymbol{V}'} \end{vmatrix} \\ &= \left|\frac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\boldsymbol{U}'}\right| \left|\frac{\partial\widehat{\boldsymbol{\Sigma}}}{\partial\boldsymbol{V}'}\right| \\ &= J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{U}} J_{\widehat{\boldsymbol{\Sigma}}\to\boldsymbol{V}}. \end{aligned}$$

Using Lemmas 4.28 and 4.30,

$$\begin{aligned} J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{U}} &= \left(\left|\frac{n-k}{a+p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2} + \boldsymbol{B}^{-1}\right|^{-1}\right)^{p} \\ &= \frac{1}{\left|\frac{n-k}{a+p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2} + \boldsymbol{B}^{-1}\right|^{p}} \end{aligned}$$

and

$$\begin{aligned} J_{\widehat{\boldsymbol{\Sigma}}\to\boldsymbol{V}} &= \left|(a+p-1)^{1/2}\,\boldsymbol{B}^{1/2}\right|^{-(p+1)} \\ &= \frac{1}{|(a+p-1)\,\boldsymbol{B}|^{(p+1)/2}} \end{aligned}$$

Thus

$$J_{\boldsymbol{\Sigma}^{-1},\widehat{\boldsymbol{\Sigma}}\to\boldsymbol{U},\boldsymbol{V}} = \frac{1}{|(a+p-1)\,\boldsymbol{B}|^{(p+1)/2}\left|\frac{n-k}{a+p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2} + \boldsymbol{B}^{-1}\right|^{p}} \tag{4.66}$$

Applying standard change of variables with the Jacobian in Equation (4.66) gives

$$L(\boldsymbol{U},\boldsymbol{V}) = c_1 \frac{\left|(a+p-1)^{-1}\,\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2}\right|^{(n-k-p-1)/2}}{\left|(a+p-1)^{-1}\,\boldsymbol{B}^{-1/2}\left((n-k)\,\boldsymbol{V}+(a+p-1)\,\boldsymbol{I}_p\right)\boldsymbol{B}^{-1/2}\boldsymbol{U}^{-1}\right|^{(n-k+a+p-1)/2}} \exp\left(-\frac{1}{2}\operatorname{Tr}\boldsymbol{U}\right) \cdot \frac{1}{|(a+p-1)\,\boldsymbol{B}|^{(p+1)/2}\left|\frac{n-k}{a+p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2}+\boldsymbol{B}^{-1}\right|^{p}}$$

This is equivalent to

$$L(\boldsymbol{U},\boldsymbol{V}) = c_2 \frac{|\boldsymbol{V}|^{(n-k-p-1)/2}}{\left|\frac{n-k}{a+p-1}\boldsymbol{V}+\boldsymbol{I}_p\right|^{(n-k+a+p-1)/2}\left|\boldsymbol{U}^{-1}\right|^{(n-k+a+p-1)/2}} \exp\left(-\frac{1}{2}\operatorname{Tr}\boldsymbol{U}\right) \tag{4.67}$$

where

$$c_2 = c_1 \frac{\left|(a+p-1)^{-1}\,\boldsymbol{B}^{-1}\right|^{n-k-p-1}}{\left|\boldsymbol{B}^{-1}\right|^{n-k+a+p-1}}$$

is a constant. Equation (4.67) separates into

$$L(\boldsymbol{U},\boldsymbol{V}) = c_2 \frac{|\boldsymbol{V}|^{(n-k-p-1)/2}}{\left|\frac{n-k}{a+p-1}\boldsymbol{V}+\boldsymbol{I}_p\right|^{(n-k+a+p-1)/2}} \cdot |\boldsymbol{U}|^{(n-k+a+p-1)/2} \exp\left(-\frac{1}{2}\operatorname{Tr}\boldsymbol{U}\right)$$

Using the definition of the multivariate F and Wishart distributions,

$$L(\boldsymbol{U},\boldsymbol{V}) \propto F_p\left(n-k, a+p-1; \tfrac{a+p-1}{n-k}\boldsymbol{I}_p\right)(\boldsymbol{V}) \cdot W_p\left(\boldsymbol{I}_p, n-k+a\right)(\boldsymbol{U}). \tag{4.68}$$

Thus $\boldsymbol{\Sigma}^{-1}\left((n-k)\,\widehat{\boldsymbol{\Sigma}}+\boldsymbol{B}^{-1}\right) = \boldsymbol{\Sigma}^{-1}\left(\widehat{\boldsymbol{S}\boldsymbol{S}}+\boldsymbol{B}^{-1}\right) \sim W_p\left(\boldsymbol{I}_p, n-k+a\right)$ and $(a+p-1)\,\boldsymbol{B}^{1/2}\widehat{\boldsymbol{\Sigma}}\boldsymbol{B}^{1/2} \sim F_p\left(n-k, a+p-1; \frac{a+p-1}{n-k}\boldsymbol{I}_p\right)$. □

In standard multivariate GLM theory (e.g., Johnson and Wichern, 2002, sections 6.4 and 7.7), without any assumptions on the population covariance matrix $\boldsymbol{\Sigma}$, the maximum likelihood estimators of $\boldsymbol{\beta}$ under H_0 and H_1 are $\widehat{\widehat{\boldsymbol{\beta}}}$ and $\widehat{\boldsymbol{\beta}}$, as noted in Equations (4.47) and (4.46) respectively. Similarly, under standard GLM theory, the maximum likelihood estimators of $\boldsymbol{\Sigma}$ under H_0 and H_1

are $\frac{1}{n}\widehat{\widehat{\boldsymbol{SS}}}$ and $\frac{1}{n}\widehat{\boldsymbol{SS}}$ respectively. This leads to the likelihood ratio test

$$\Lambda = \left(\frac{\left|\frac{1}{n}\widehat{\boldsymbol{SS}}\right|}{\left|\frac{1}{n}\widehat{\widehat{\boldsymbol{SS}}}\right|} \right)^{n/2} = \left(\frac{\left|\widehat{\boldsymbol{SS}}\right|}{\left|\widehat{\widehat{\boldsymbol{SS}}}\right|} \right)^{n/2}$$

for testing the hypothesis $H_0 : \beta \in \omega$ versus $H_1 : \beta \in \mathbb{R}^k$.

In standard GLM theory, the sums of squares and crossproducts $\widehat{\boldsymbol{SS}} \sim W_p(n-k, \boldsymbol{\Sigma})$ under H_0. The extra sums of squares and crossproducts, defined as the difference between the models fit under H_0 and H_1, equals $\widehat{\widehat{\boldsymbol{SS}}} - \widehat{\boldsymbol{SS}}$ and is distributed as $W_p(k-r, \boldsymbol{\Sigma})$ by Cochran's Theorem (Cochran, 1934). Furthermore, by Craig's Theorem (Craig, 1943), $\widehat{\widehat{\boldsymbol{SS}}} - \widehat{\boldsymbol{SS}}$ is independent of $\widehat{\boldsymbol{SS}}$. Thus, $\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\boldsymbol{SS}}\right)\boldsymbol{\Sigma}^{-1/2} \sim W_p\left(n-k, \boldsymbol{I}_p\right)$ and $\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\widehat{\boldsymbol{SS}}} - \widehat{\boldsymbol{SS}}\right)\boldsymbol{\Sigma}^{-1/2} \sim W_p\left(k-r, \boldsymbol{I}_p\right)$. Using the definition of Wilks' lambda,

$$\frac{\left|\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\boldsymbol{SS}}\right)\boldsymbol{\Sigma}^{-1/2}\right|}{\left|\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\boldsymbol{SS}}\right)\boldsymbol{\Sigma}^{-1/2} + \boldsymbol{\Sigma}^{-1/2}\left(\widehat{\widehat{\boldsymbol{SS}}} - \widehat{\boldsymbol{SS}}\right)\boldsymbol{\Sigma}^{-1/2}\right|} = \frac{\left|\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\boldsymbol{SS}}\right)\boldsymbol{\Sigma}^{-1/2}\right|}{\left|\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\widehat{\boldsymbol{SS}}}\right)\boldsymbol{\Sigma}^{-1/2}\right|} \sim \Lambda(p, n-k, k-r)$$

Since

$$\frac{\left|\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\boldsymbol{SS}}\right)\boldsymbol{\Sigma}^{-1/2}\right|}{\left|\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\widehat{\boldsymbol{SS}}}\right)\boldsymbol{\Sigma}^{-1/2}\right|} = \frac{\left|\boldsymbol{\Sigma}^{-1/2}\right|\left|\widehat{\boldsymbol{SS}}\right|\left|\boldsymbol{\Sigma}^{-1/2}\right|}{\left|\boldsymbol{\Sigma}^{-1/2}\right|\left|\widehat{\widehat{\boldsymbol{SS}}}\right|\left|\boldsymbol{\Sigma}^{-1/2}\right|}$$

the distribution of the likelihood ratio statistic can be found using the relationship

$$\Lambda^{2/n} = \frac{\left|\widehat{\boldsymbol{SS}}\right|}{\left|\widehat{\widehat{\boldsymbol{SS}}}\right|} \sim \Lambda(p, n-k, k-r)$$

with the cutoff value chosen to achieve the desired level of significance. Alternatively, since Λ is a likelihood ratio statistic, it follows that $-2\log\Lambda = -2\left(\log\widehat{\boldsymbol{SS}} - \log\widehat{\widehat{\boldsymbol{SS}}}\right) \sim \chi^2_{k-r}$. The cutoff value for $-2\log\Lambda$ would also be chosen to achieve the desired level of significance.

The results of Theorems 4.48 and 4.49 lead to modifications in standard multivariate GLM

theory for hypothesis tests under the RVM assumptions. Establishing the modified hypothesis tests requires the distribution of the quantities corresponding to the statistics in standard GLM theory. Theorem 4.49 shows that $\boldsymbol{\Sigma}^{-1}\left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right) \sim W_p\left(\boldsymbol{I}_p, n-k+a\right)$; an analogous argument establishes that $\boldsymbol{\Sigma}^{-1}\left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right) \sim W_p\left(\boldsymbol{I}_p, n-r+a\right)$. The extra sums of squares and crossproducts under the RVM assumptions equals $\left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right) - \left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right) = \widehat{\widehat{\boldsymbol{SS}}} - \widehat{\boldsymbol{SS}}$, so that the extra sums of squares and crossproducts is distributed as $W_p(k-r, \boldsymbol{\Sigma})$ under RVM. Consequently, $\boldsymbol{\Sigma}^{-1/2}\left[\left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right) - \left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right)\right]\boldsymbol{\Sigma}^{-1/2} = \boldsymbol{\Sigma}^{-1/2}\left[\widehat{\widehat{\boldsymbol{SS}}} - \widehat{\boldsymbol{SS}}\right]\boldsymbol{\Sigma}^{-1/2} \sim W_p\left(k-r, \boldsymbol{I}_p\right)$. The extra sums of squares and crossproducts is also independent of $\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}$, since the latter quantity depends on the data only through $\widehat{\boldsymbol{SS}}$.

Unfortunately, under RVM, the likelihood ratio test $\widetilde{\Lambda}$ given in Equation (4.48) no longer has a Wilks' lambda distribution. Note that $\widetilde{\Lambda}$ may be expressed as

$$\begin{aligned}
\widetilde{\Lambda}^{(n+a-p-1)/2} &= \frac{\left|\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right|}{\left|\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right|} \\
&= \frac{\left|\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right|}{\left|\left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right) + \left[\widehat{\widehat{\boldsymbol{SS}}} - \widehat{\boldsymbol{SS}}\right]\right|} \\
&= \frac{\left|\boldsymbol{\Sigma}^{-1/2}\right|\left|\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right|\left|\boldsymbol{\Sigma}^{-1/2}\right|}{\left|\boldsymbol{\Sigma}^{-1/2}\right|\left|\left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right) + \left[\widehat{\widehat{\boldsymbol{SS}}} - \widehat{\boldsymbol{SS}}\right]\right|\left|\boldsymbol{\Sigma}^{-1/2}\right|} \\
&= \frac{\left|\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{-1/2}\right|}{\left|\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{-1/2} + \boldsymbol{\Sigma}^{-1/2}\left[\widehat{\widehat{\boldsymbol{SS}}} - \widehat{\boldsymbol{SS}}\right]\boldsymbol{\Sigma}^{-1/2}\right|}
\end{aligned} \tag{4.69}$$

The determinant $\left|\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{-1/2}\right|$ in the numerator of Equation (4.69) is easily rearranged to $\left|\boldsymbol{\Sigma}^{-1}\left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right)\right|$. However, no such arrangement of the quantity $\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{-1/2}$ in the denominator is possible. Since $\boldsymbol{\Sigma}^{-1/2}\left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{-1/2}$ does not follow a standard Wishart distribution, $\widetilde{\Lambda}$ does not meet the conditions of Wilks' lambda in Definition 4.47 under this approach.

A similar problem arises if the Cholesky decomposition is used. Let $\boldsymbol{\Sigma}^{-1}$ be decomposed as $\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Sigma}^{*}\boldsymbol{\Sigma}^{*\prime}$, where $\boldsymbol{\Sigma}^{*}$ is a lower triangular matrix. Then $\boldsymbol{\Sigma}^{*\prime}\left[\widehat{\widehat{\boldsymbol{SS}}} - \widehat{\boldsymbol{SS}}\right]\boldsymbol{\Sigma}^{*} \sim W_p\left(\boldsymbol{I}_p, k-r\right)$. The

likelihood ratio test becomes

$$\begin{aligned}
\widetilde{\Lambda}^{(n+a-p-1)/2} &= \frac{\left|\widehat{SS} + \boldsymbol{B}^{-1}\right|}{\left|\widehat{\widehat{SS}} + \boldsymbol{B}^{-1}\right|} \\
&= \frac{\left|\widehat{SS} + \boldsymbol{B}^{-1}\right|}{\left|\left(\widehat{SS} + \boldsymbol{B}^{-1}\right) + \left[\widehat{\widehat{SS}} - \widehat{SS}\right]\right|} \\
&= \frac{|\boldsymbol{\Sigma}^{*\prime}|\left|\widehat{SS} + \boldsymbol{B}^{-1}\right||\boldsymbol{\Sigma}^{*}|}{|\boldsymbol{\Sigma}^{*\prime}|\left|\left(\widehat{SS} + \boldsymbol{B}^{-1}\right) + \left[\widehat{\widehat{SS}} - \widehat{SS}\right]\right||\boldsymbol{\Sigma}^{*}|} \\
&= \frac{\left|\boldsymbol{\Sigma}^{*\prime}\left(\widehat{SS} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{*}\right|}{\left|\boldsymbol{\Sigma}^{*\prime}\left(\widehat{SS} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{*} + \boldsymbol{\Sigma}^{*\prime}\left[\widehat{\widehat{SS}} - \widehat{SS}\right]\boldsymbol{\Sigma}^{*}\right|}
\end{aligned} \tag{4.70}$$

Again the determinant in the numerator is easily rearranged to $\left|\boldsymbol{\Sigma}^{-1}\left(\widehat{SS} + \boldsymbol{B}^{-1}\right)\right|$, but no such arrangement of the quantity $\boldsymbol{\Sigma}^{*\prime}\left(\widehat{\widehat{SS}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{*}$ in the denominator is possible. As $\boldsymbol{\Sigma}^{*\prime}\left(\widehat{\widehat{SS}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{*}$ does not follow a standard Wishart distribution, $\widetilde{\Lambda}$ does not meet the conditions of Wilks' lambda in Definition 4.47 under this approach either.

An alternative is to consider

$$\begin{aligned}
\widetilde{\Lambda}^{(n+a-p-1)/2} &= \frac{\left|\boldsymbol{\Sigma}^{-1}\right|\left|\widehat{SS} + \boldsymbol{B}^{-1}\right|}{\left|\boldsymbol{\Sigma}^{-1}\right|\left|\left(\widehat{SS} + \boldsymbol{B}^{-1}\right) + \left[\widehat{\widehat{SS}} - \widehat{SS}\right]\right|} \\
&= \frac{\left|\boldsymbol{\Sigma}^{-1}\left(\widehat{SS} + \boldsymbol{B}^{-1}\right)\right|}{\left|\boldsymbol{\Sigma}^{-1}\left(\widehat{SS} + \boldsymbol{B}^{-1}\right) + \boldsymbol{\Sigma}^{-1}\left[\widehat{\widehat{SS}} - \widehat{SS}\right]\right|}
\end{aligned} \tag{4.71}$$

However, note that it is the distribution of $\boldsymbol{\Sigma}^{-1/2}\left[\widehat{\widehat{SS}} - \widehat{SS}\right]\boldsymbol{\Sigma}^{-1/2}$ that is $W_p\left(\boldsymbol{I}_p, k-r\right)$; the distribution of $\boldsymbol{\Sigma}^{-1}\left[\widehat{\widehat{SS}} - \widehat{SS}\right]$ is $\left|\boldsymbol{\Sigma}^{-1}\right| W_p\left(\boldsymbol{I}_p, k-r\right)$. Under this approach, $\widetilde{\Lambda}$ still does not meet the conditions of the Wilks' lambda in Definition 4.47 and does not follow this distribution.

Another alternative is to reformulate the transformations given in Equations (4.64) and (4.65) so that the determinant in Equation (4.69) or Equation (4.70) may be evaluated. Beginning with

the joint likelihood in Equation (4.63), which is

$$\begin{aligned} L\left(\widehat{\boldsymbol{\Sigma}}, \boldsymbol{\Sigma}^{-1}\right) &= c_1 \frac{\left|\widehat{\boldsymbol{\Sigma}}\right|^{(n-k-p-1)/2}}{|\boldsymbol{\Sigma}|^{(n-k+a-p-1)/2}} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)\right]\right) \\ &= c_1 \frac{\left|\widehat{\boldsymbol{\Sigma}}\right|^{(n-k-p-1)/2}}{|\boldsymbol{\Sigma}|^{(n-k+a-p-1)/2}} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1/2}\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{-1/2}\right]\right), \end{aligned}$$

consider the transformations

$$\boldsymbol{U} = \boldsymbol{\Sigma}^{-1/2}\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{-1/2}$$

and

$$\boldsymbol{V} = (a-p-1)^{1/2}\,\boldsymbol{B}^{1/2}\widehat{\boldsymbol{\Sigma}}\boldsymbol{B}^{1/2}\,(a-p-1)^{1/2}\,.$$

The Jacobian of the transformation remains $J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{U}} J_{\widehat{\boldsymbol{\Sigma}}\to\boldsymbol{V}}$, but now $J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{U}}$ is given by

$$\begin{aligned} J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{U}} &= J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{\Sigma}^{-1/2}} J_{\boldsymbol{\Sigma}^{-1/2}\to\boldsymbol{U}} \\ &= J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{\Sigma}^{-1/2}} \frac{1}{J_{\boldsymbol{U}\to\boldsymbol{\Sigma}^{-1/2}}} \end{aligned}$$

using Properties (1) and (2) of Jacobians. Using Lemma 4.29,

$$J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{\Sigma}^{-1/2}} = \prod_{i\le j}^{p}\left(\lambda_i + \lambda_j\right)$$

where $\lambda_1 \ge \lambda_2 \ge \cdots \ge \lambda_p$ are the ordered eigenvalues of $\boldsymbol{\Sigma}^{1/2}$. Using Lemma 4.30,

$$J_{\boldsymbol{U}\to\boldsymbol{\Sigma}^{-1/2}} = \prod_{i\le j}^{p}\left(\eta_i + \eta_j\right)$$

where $\eta_1 \geq \eta_2 \geq \cdots \geq \eta_p$ are the ordered eigenvalues of $\boldsymbol{\Sigma}^{-1/2}\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)$. Thus

$$J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{U}} = \frac{\prod_{i\leq j}^{p}\left(\lambda_i+\lambda_j\right)}{\prod_{i\leq j}^{p}\left(\eta_i+\eta_j\right)}.$$

To derive the joint likelihood of $\boldsymbol{U}$ and $\boldsymbol{V}$, note that

$$\begin{aligned}
|\boldsymbol{U}| &= \left|\boldsymbol{\Sigma}^{-1/2}\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{-1/2}\right| \\
&= \left|\boldsymbol{\Sigma}^{-1/2}\right|\left|(n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right|\left|\boldsymbol{\Sigma}^{-1/2}\right| \\
&= \left|\boldsymbol{\Sigma}^{-1/2}\right|\left|\boldsymbol{\Sigma}^{-1/2}\right|\left|(n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right| \\
&= \left|\boldsymbol{\Sigma}^{-1}\right|\left|(n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right|,
\end{aligned}$$

so that

$$\left|\boldsymbol{\Sigma}^{-1}\right| = \left|(n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right|^{-1}|\boldsymbol{U}|\,.$$

Thus the likelihood is

$$\begin{aligned}
L(\boldsymbol{U},\boldsymbol{V}) = c_1 &\left|(a-p-1)^{-1}\,\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2}\right|^{(n-k-p-1)/2}\left|\frac{n-k}{a-p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2} + \boldsymbol{B}^{-1}\right|^{-(n-k+a-p-1)/2} \\
&\cdot |\boldsymbol{U}|^{(n-k+a-p-1)/2}\exp\left(-\frac{1}{2}\operatorname{Tr}\boldsymbol{U}\right)\cdot\frac{\prod_{i\leq j}^{p}\left(\lambda_i+\lambda_j\right)}{\prod_{i\leq j}^{p}\left(\eta_i+\eta_j\right)}
\end{aligned}$$

This is equivalent to

$$\begin{aligned}
L(\boldsymbol{U},\boldsymbol{V}) &= c_2\frac{|\boldsymbol{V}|^{(n-k-p-1)/2}}{\left|\frac{n-k}{a-p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2} + \boldsymbol{B}^{-1}\right|^{(n-k+a-p-1)/2}}|\boldsymbol{U}|^{(n-k+a-p-1)/2}\exp\left(-\frac{1}{2}\operatorname{Tr}\boldsymbol{U}\right)\frac{\prod_{i\leq j}^{p}\left(\lambda_i+\lambda_j\right)}{\prod_{i\leq j}^{p}\left(\eta_i+\eta_j\right)} \\
&= c_3\frac{|\boldsymbol{V}|^{(n-k-p-1)/2}}{\left|\frac{n-k}{a-p-1}\boldsymbol{V} + \boldsymbol{I}_p\right|^{(n-k+a-p-1)/2}}|\boldsymbol{U}|^{(n-k+a-p-1)/2}\exp\left(-\frac{1}{2}\operatorname{Tr}\boldsymbol{U}\right)\frac{\prod_{i\leq j}^{p}\left(\lambda_i+\lambda_j\right)}{\prod_{i\leq j}^{p}\left(\eta_i+\eta_j\right)} \qquad (4.72)
\end{aligned}$$

where

$$c_2 = c_1 \frac{1}{|(a-p-1)\,\boldsymbol{B}|^{(n-k-p-1)/2}}$$

and

$$c_3 = c_2 \frac{1}{|\boldsymbol{B}|^{(n-k+a-p-1)/2}}$$

are constants. The use of Equation (4.72) in the RVM method would then depend on the factorization into two separate functions, with one being a function of $\boldsymbol{U}$ only and the other being a function of $\boldsymbol{V}$ only. However, Olkin and Rubin (1964) show that the term $\frac{\prod_{i\leq j}^{p}(\lambda_i+\lambda_j)}{\prod_{i\leq j}^{p}(\eta_i+\eta_j)}$ generally cannot be factorized in this manner, providing a counterexample when $p = 2$. The term $\frac{\prod_{i\leq j}^{p}(\lambda_i+\lambda_j)}{\prod_{i\leq j}^{p}(\eta_i+\eta_j)}$ cannot be manipulated to eliminate the $\boldsymbol{\Sigma}^{-1/2}$ term and produce a function of $\boldsymbol{U}$ only. A more general approach is to use the results of Equations (4.27) and (4.29) to obtain

$$\begin{aligned}
\frac{\prod_{i\leq j}^{p}\left(\lambda_i+\lambda_j\right)}{\prod_{i\leq j}^{p}\left(\eta_i+\eta_j\right)} &= \frac{\left|\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{\Sigma}^{-1/2}+\boldsymbol{\Sigma}^{-1/2}\otimes\boldsymbol{I}_p\right)\boldsymbol{D}_p\right|}{\left|\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{\Sigma}^{-1/2}\left[(n-k)\,\widehat{\boldsymbol{\Sigma}}+\boldsymbol{B}^{-1}\right]+\boldsymbol{\Sigma}^{-1/2}\left[(n-k)\,\widehat{\boldsymbol{\Sigma}}+\boldsymbol{B}^{-1}\right]\otimes\boldsymbol{I}_p\right)\boldsymbol{D}_p\right|} \\
&= \left|\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{\Sigma}^{-1/2}+\boldsymbol{\Sigma}^{-1/2}\otimes\boldsymbol{I}_p\right)\boldsymbol{D}_p\right| \\
&\quad\cdot\left|\boldsymbol{L}_p\left(\boldsymbol{I}_p\otimes\boldsymbol{\Sigma}^{-1/2}\left[(n-k)\,\widehat{\boldsymbol{\Sigma}}+\boldsymbol{B}^{-1}\right]+\boldsymbol{\Sigma}^{-1/2}\left[(n-k)\,\widehat{\boldsymbol{\Sigma}}+\boldsymbol{B}^{-1}\right]\otimes\boldsymbol{I}_p\right)\boldsymbol{D}_p\right|^{-1}
\end{aligned}$$

Using Lemma 4.14 and Property (10) of the elimination and duplication matrices gives

$$
\begin{aligned}
&\left|\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{\Sigma}^{-1/2} + \boldsymbol{\Sigma}^{-1/2} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{\Sigma}^{-1/2}\left[(n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right] + \boldsymbol{\Sigma}^{-1/2}\left[(n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right]\right.\right. \\
&\quad \left.\left. \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p\right|^{-1} \\
&= \left|\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{\Sigma}^{-1/2} + \boldsymbol{\Sigma}^{-1/2} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p\right|\left|\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{\Sigma}^{-1/2}\left[(n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right] + \boldsymbol{\Sigma}^{-1/2}\left[(n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right]\right.\right. \\
&\quad \left.\left. \otimes \boldsymbol{I}_p\right)^{-1}\boldsymbol{D}_p\right| \\
&= \left|\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{\Sigma}^{-1/2} + \boldsymbol{\Sigma}^{-1/2} \otimes \boldsymbol{I}_p\right)\boldsymbol{D}_p\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{\Sigma}^{-1/2}\left[(n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right] + \boldsymbol{\Sigma}^{-1/2}\left[(n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right]\right.\right. \\
&\quad \left.\left. \otimes \boldsymbol{I}_p\right)^{-1}\boldsymbol{D}_p\right| \\
&= \left|\boldsymbol{L}_p\left(\boldsymbol{I}_p \otimes \boldsymbol{\Sigma}^{-1/2} + \boldsymbol{\Sigma}^{-1/2} \otimes \boldsymbol{I}_p\right)\left(\boldsymbol{I}_p \otimes \boldsymbol{\Sigma}^{-1/2}\left[(n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right] + \boldsymbol{\Sigma}^{-1/2}\left[(n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right]\right.\right. \\
&\quad \left.\left. \otimes \boldsymbol{I}_p\right)^{-1}\boldsymbol{D}_p\right|
\end{aligned}
$$

so that the factorization depends on $\left(\boldsymbol{I}_p \otimes \boldsymbol{\Sigma}^{-1/2}\left[(n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right] + \boldsymbol{\Sigma}^{-1/2}\left[(n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right] \otimes \boldsymbol{I}_p\right)$ being separated into $\left(\boldsymbol{I}_p \otimes \boldsymbol{\Sigma}^{-1/2} + \boldsymbol{\Sigma}^{-1/2} \otimes \boldsymbol{I}_p\right)$ and a function of $(n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}$ only, which generally cannot be done. Thus, the transformation $\boldsymbol{U} = \boldsymbol{\Sigma}^{-1/2}\left((n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{-1/2}$ does not lead to a suitable formulation of the RVM method.

A similar difficulty arises if the Cholesky decomposition is used instead of the square root decomposition. Let $\boldsymbol{\Sigma}^{-1}$ be decomposed as $\boldsymbol{\Sigma}^{-1} = \boldsymbol{\Sigma}^{*}\boldsymbol{\Sigma}^{*\prime}$, where Σ^{*} is a lower triangular matrix. Beginning with the joint likelihood in Equation (4.63), which is

$$
\begin{aligned}
L\left(\widehat{\boldsymbol{\Sigma}}, \boldsymbol{\Sigma}^{-1}\right) &= c_1 \frac{\left|\widehat{\boldsymbol{\Sigma}}\right|^{(n-k-p-1)/2}}{|\boldsymbol{\Sigma}|^{(n-k+a-p-1)/2}} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}\left((n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)\right]\right) \\
&= c_1 \frac{\left|\widehat{\boldsymbol{\Sigma}}\right|^{(n-k-p-1)/2}}{|\boldsymbol{\Sigma}|^{(n-k+a-p-1)/2}} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{*}\boldsymbol{\Sigma}^{*\prime}\left((n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)\right]\right) \\
&= c_1 \frac{\left|\widehat{\boldsymbol{\Sigma}}\right|^{(n-k-p-1)/2}}{|\boldsymbol{\Sigma}|^{(n-k+a-p-1)/2}} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{*\prime}\left((n-k)\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{*}\right]\right),
\end{aligned}
$$

consider the transformations

$$\boldsymbol{U} = \boldsymbol{\Sigma}^{*\prime}\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)\boldsymbol{\Sigma}^{*}$$

and

$$\boldsymbol{V} = (a-p-1)^{1/2}\,\boldsymbol{B}^{1/2}\widehat{\boldsymbol{\Sigma}}\boldsymbol{B}^{1/2}\,(a-p-1)^{1/2}\,.$$

The Jacobian of the transformation remains $J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{U}}J_{\widehat{\boldsymbol{\Sigma}}\to\boldsymbol{V}}$, but now $J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{U}}$ is given by

$$\begin{aligned} J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{U}} &= J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{\Sigma}^{*}}J_{\boldsymbol{\Sigma}^{*}\to\boldsymbol{U}} \\ &= J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{\Sigma}^{*}}\frac{1}{J_{\boldsymbol{U}\to\boldsymbol{\Sigma}^{*}}} \end{aligned}$$

using Properties (1) and (2) of Jacobians. Using Lemma 4.31,

$$J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{\Sigma}^{*}} = 2^{p}\prod_{i=1}^{p}\sigma_{ii}^{p-i+1}$$

where $\sigma_{ii}, i = 1, 2, \ldots, p$ are the diagonal elements of $\boldsymbol{\Sigma}^{*}$. Using Lemma 4.31,

$$J_{\boldsymbol{U}\to\boldsymbol{\Sigma}^{*}} = 2^{p}\prod_{i=1}^{p}\sigma_{ii}^{i}\left|\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)_{[i]}\right|$$

Thus

$$\begin{aligned} J_{\boldsymbol{\Sigma}^{*}\to\boldsymbol{U}} &= \frac{2^{p}\prod_{i=1}^{p}\sigma_{ii}^{p-i+1}}{2^{p}\prod_{i=1}^{p}\sigma_{ii}^{i}\left|\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)_{[i]}\right|} \\ &= \frac{\prod_{i=1}^{p}\sigma_{ii}^{p-2i+1}}{\prod_{i=1}^{p}\left|\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)_{[i]}\right|}. \end{aligned}$$

Since σ_{ii} are the diagonal elements of $\boldsymbol{\Sigma}^{*}$, it is clear that the use of the Cholesky decomposition will not lead to a transformation in terms of $\boldsymbol{U}$ and $\boldsymbol{V}$ only. Thus, the transformation

$\boldsymbol{U} = \boldsymbol{\Sigma}^* \left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1} \right) \boldsymbol{\Sigma}^*$ does not lead to a suitable formulation of the RVM method.

Although an exact distribution for $\widetilde{\Lambda}$ is not available, since it is a likelihood ratio test, the value of $-2 \log \widetilde{\Lambda}$ approximately follows a χ^2 distribution and may be used for hypothesis testing. This is an asymptotic approximation, so the accuracy of the test may be compromised for small sample sizes. The likelihood ratio test in Equation (4.48) may be written as

$$\begin{aligned}
\widetilde{\Lambda}^{(n+a-p-1)/2} &= \frac{\left| \widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1} \right|}{\left| \widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1} \right|} \\
&= \frac{\left| \boldsymbol{\Sigma}^{-1} \right| \left| \widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1} \right|}{\left| \boldsymbol{\Sigma}^{-1} \right| \left| \widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1} \right|} \\
&= \frac{\left| \boldsymbol{\Sigma}^{-1} \left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1} \right) \right|}{\left| \boldsymbol{\Sigma}^{-1} \left(\widehat{\widehat{\boldsymbol{SS}}} + \boldsymbol{B}^{-1} \right) \right|}.
\end{aligned}$$

The distributional results in Equation (4.68) show that $\boldsymbol{\Sigma}^{-1} \left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1} \right) \sim W_p \left(\boldsymbol{I}_p, n-k+a \right)$ and, by extension, that $\boldsymbol{\Sigma}^{-1} \left((n-k)\,\widehat{\widehat{\boldsymbol{\Sigma}}} + \boldsymbol{B}^{-1} \right) \sim W_p \left(\boldsymbol{I}_p, n-r+a \right)$. This implies that the degrees of freedom for the χ^2 test are $k-r$. Using these results, the multivariate RVM may be applied to a wide variety of GLMs in a similar manner to the univariate RVM. For the multivariate RVM, the ratio of the generalized variances under the null and alternative hypotheses are used in conducting the hypothesis tests, and both the numerator and denominator sums of squares and degrees of freedom are adjusted. The hyperparameters a and $\boldsymbol{B}$ may be estimated from $\widehat{\boldsymbol{\Sigma}}$, the residuals from the standard GLM without any assumptions on the distribution of $\boldsymbol{\Sigma}^{-1}$, using a numerical maximization routine. However, previous studies have noted problems of identifiability when estimating an unstructured prior for the Wishart distribution, even when numerical optimization routines report convergence to a solution (Le *et al.*, 1998). To avoid such problems in the multivariate RVM method, structure may be imposed on the $\boldsymbol{B}$ matrix to reduce the number of parameters estimated. A compound symmetric structure is a reasonable choice for microarray studies based on the assumed structure of the sample variance-covariance matrix as well as the previous work by Archer *et al.* (2006). Once estimates of the hyperparameters are obtained, the

value of $-2\log\widetilde{\Lambda}$ may be computed and compared to the χ^2_{k-r} distribution with the appropriate cutoff value.

The multivariate RVM method may be applied to class comparison problems for determining the significance of gene expression changes between conditions. For such a parameterization, k would represent the number of conditions, and the columns of $\boldsymbol{X}$ would be indicator variables denoting the condition of each chip in the experiment. The reduced model would correspond to the hypothesis that $k - r$ of the conditions are equivalent, with the corresponding reduction in the dimension of $\boldsymbol{X}$. The class comparison problem then becomes a multivariate analysis of variance (MANOVA) problem with adjustment in the residuals and degrees of freedom for the test statistic based on the RVM assumptions. Thus, although the multivariate RVM method is not currently implemented in software packages for microarray data analysis, hypothesis tests based on multivariate RVM may be carried out using standard packages for MANOVA or multivariate regression.

4.3.5 Modified Random Variance Model for Singular Covariance Matrices

The preceding formulation of the multivariate RVM method relies on the sample covariance matrix $\widehat{\boldsymbol{\Sigma}}$ (or, equivalently, the samples sums of squares and crossproducts $\widehat{\boldsymbol{SS}}$) being nonsingular and of full rank. These conditions require that $n - k > p$; otherwise, the Wishart density function given in Definition 4.34 does not exist. The requirement that $n-k > p$ may be unrealistic for most microarray studies; despite efforts to increase the sample size for microarray experiments, many studies still use only a small number of chips (Jain *et al.*, 2003). For example, the most common probeset size on the human U95 and U133 GeneChips is 11 probes. Assuming a two class comparison, 7 chips would be required in each class to achieve an adequate number of samples. Other types of chips, such as the *Drosophilia* DrosGenome1 chip with 14 probes per probeset, would require an even larger number of replicates. However, the requirement that $n - k > p$ may be removed by utilizing the pseudo-Wishart distribution in Definition 4.35 when $n - k \leq p$. However, based on Lemma 4.36, a different computational formula would need to be used in

software implementations to account for the differences in the nonsingular and pseudo-Wishart distributions.

Theorem 4.48 is unaltered by the sample covariance matrix $\widehat{\boldsymbol{\Sigma}}$ being singular, with the assumption that $\boldsymbol{B}$ and $\boldsymbol{\Sigma}$ are of full rank. Thus, the same likelihood ratio test may be used for hypothesis testing in the singular case as in the nonsingular. However, modifications to Theorem 4.49 are necessary to incorporate the pseudo-Wishart distribution of $\widehat{\boldsymbol{\Sigma}}$. These modifications lead to the following theorem.

Theorem 4.50. *Assume that $n - k \leq p$, so that $\widehat{\boldsymbol{\Sigma}}$ is singular. Then, for $\widehat{\boldsymbol{\Sigma}}$ and $\widetilde{\boldsymbol{\Sigma}}$ defined as in nonsingular multivariate RVM,*

$$\boldsymbol{\Sigma}^{-1}\left((n-k+a)\,\widetilde{\boldsymbol{\Sigma}}\right) = \boldsymbol{\Sigma}^{-1}\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right) \sim W_p\left(\boldsymbol{I}_p, n-k+a\right)$$

and

$$\begin{aligned}(a+p-1)\,\boldsymbol{B}^{-1/2}\widehat{\boldsymbol{\Sigma}}\boldsymbol{B}^{-1/2} &\sim F_p^{n-k}\left(n-k, a+p-1; \tfrac{a+p-1}{n-k} I_p\right) \\ &\equiv M\beta_{II}^{n-k}\left(p, n-k, a+p-1; \tfrac{n-k}{a+p-1} I_p\right)\end{aligned}$$

Proof. From singular GLM theory, $(n-k)\,\widehat{\boldsymbol{\Sigma}} \sim W_p(n-k, \boldsymbol{\Sigma})$. Since $n-k < p$ by assumption, $(n-k)\,\widehat{\boldsymbol{\Sigma}}$ has a pseudo-Wishart distribution. Let $\boldsymbol{L}_1 = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_{n-k})$, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n-k}$ are the first $n-k$ ordered eigenvalues of $(n-k)\,\widehat{\boldsymbol{\Sigma}}$, and let $\boldsymbol{E}_1$ be the matrix of the corresponding eigenvectors, so that $\boldsymbol{E}_1\boldsymbol{E}_1' = \boldsymbol{I}_{n-k}$. Then the density of $(n-k)\,\widehat{\boldsymbol{\Sigma}}$ is given by

$$f\left((n-k)\,\widehat{\boldsymbol{\Sigma}}\right) = \frac{\pi^{\left(-p(n-k)+(n-k)^2\right)/2}\,|\boldsymbol{L}_1|^{(n-k-p-1)/2}\left|\boldsymbol{\Sigma}^{-1}\right|^{(n-k)/2}}{2^{(n-k)p/2}\Gamma_{n-k}\left(\frac{n-k}{2}\right)} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}(n-k)\,\widehat{\boldsymbol{\Sigma}}\right]\right)$$

Under RVM, $\boldsymbol{\Sigma}^{-1} \sim W_p(\boldsymbol{B}, a)$, where $\boldsymbol{B}$ is assumed to be full rank. The joint density of $\widehat{\boldsymbol{\Sigma}}$ and $\boldsymbol{\Sigma}^{-1}$

is

$$L\left(\widehat{\boldsymbol{\Sigma}}, \boldsymbol{\Sigma}^{-1}\right) = \frac{\pi^{\left(-p(n-k)+(n-k)^2\right)/2}\,(n-k)\,|\boldsymbol{L}_1|^{(n-k-p-1)/2}\left|\boldsymbol{\Sigma}^{-1}\right|^{(n-k)/2}}{2^{(n-k)p/2}\Gamma_{n-k}\left(\frac{n-k}{2}\right)} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}\,(n-k)\,\widehat{\boldsymbol{\Sigma}}\right]\right)$$
$$\cdot \frac{\left|\boldsymbol{\Sigma}^{-1}\right|^{(a-p-1)/2}}{2^{ap/2}\,|\boldsymbol{B}|^{a/2}\,\Gamma_p\left(\frac{a}{2}\right)} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{B}^{-1}\boldsymbol{\Sigma}^{-1}\right]\right).$$

This simplifies to

$$L\left(\widehat{\boldsymbol{\Sigma}}, \boldsymbol{\Sigma}^{-1}\right) = \frac{\pi^{\left(-p(n-k)+(n-k)^2\right)/2}\,(n-k)}{2^{(n-k+a)p/2}\,|\boldsymbol{B}|^{a/2}\,\Gamma_{n-k}\left(\frac{n-k}{2}\right)\Gamma_p\left(\frac{a}{2}\right)} |\boldsymbol{L}_1|^{(n-k-p-1)/2}\left|\boldsymbol{\Sigma}^{-1}\right|^{(n-k+a-p-1)/2}$$
$$\cdot \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)\right]\right)$$

This is equivalent to

$$= c_1 \frac{|\boldsymbol{L}_1|^{(n-k-p-1)/2}}{|\boldsymbol{\Sigma}|^{(n-k+a-p-1)/2}} \exp\left(-\frac{1}{2}\operatorname{Tr}\left[\boldsymbol{\Sigma}^{-1}\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)\right]\right)$$

where

$$c_1 = \frac{\pi^{\left(-p(n-k)+(n-k)^2\right)/2}\,(n-k)}{2^{(n-k+a)p/2}\,|\boldsymbol{B}|^{a/2}\,\Gamma_{n-k}\left(\frac{n-k}{2}\right)\Gamma_p\left(\frac{a}{2}\right)}$$

is a constant. Let

$$\boldsymbol{U} = \boldsymbol{\Sigma}^{-1}\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)$$

and

$$\boldsymbol{V} = (a+p-1)^{1/2}\,\boldsymbol{B}^{1/2}\widehat{\boldsymbol{\Sigma}}\boldsymbol{B}^{1/2}\,(a+p-1)^{1/2}$$

so that

$$\begin{aligned}\boldsymbol{\Sigma}^{-1} &= \boldsymbol{U}\left((n-k)\,\widehat{\boldsymbol{\Sigma}} + \boldsymbol{B}^{-1}\right)^{-1} \\ &= \boldsymbol{U}\left(\frac{n-k}{a+p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2} + \boldsymbol{B}^{-1}\right)^{-1}\end{aligned}$$

and

$$\widehat{\boldsymbol{\Sigma}} = (a+p-1)^{-1/2}\,\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2}\,(a+p-1)^{-1/2}\,.$$

Let $\boldsymbol{L}_2 = \operatorname{diag}(\kappa_1, \kappa_2, \ldots, \kappa_{n-k})$, where $\kappa_1 \geq \kappa_2 \geq \cdots \geq \kappa_{n-k}$ are the first $n-k$ ordered eigenvalues of $\boldsymbol{V}$, and let $\boldsymbol{E}_2$ be the matrix of the corresponding eigenvectors, so that $\boldsymbol{E}_2\boldsymbol{E}_2' = \boldsymbol{I}_{n-k}$. The Jacobian of the transformation is

$$\begin{aligned}J_{\boldsymbol{\Sigma}^{-1},\widehat{\boldsymbol{\Sigma}}\to\boldsymbol{U},\boldsymbol{V}} &= \begin{vmatrix} \dfrac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\boldsymbol{U}'} & \dfrac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\boldsymbol{V}'} \\ \dfrac{\partial\widehat{\boldsymbol{\Sigma}}}{\partial\boldsymbol{U}'} & \dfrac{\partial\widehat{\boldsymbol{\Sigma}}}{\partial\boldsymbol{V}'} \end{vmatrix} \\ &= \begin{vmatrix} \dfrac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\boldsymbol{U}'} & \dfrac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\boldsymbol{V}'} \\ \boldsymbol{0} & \dfrac{\partial\widehat{\boldsymbol{\Sigma}}}{\partial\boldsymbol{V}'} \end{vmatrix} \\ &= \left|\frac{\partial\boldsymbol{\Sigma}^{-1}}{\partial\boldsymbol{U}'}\right| \left|\frac{\partial\widehat{\boldsymbol{\Sigma}}}{\partial\boldsymbol{V}'}\right| \\ &= J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{U}} J_{\widehat{\boldsymbol{\Sigma}}\to\boldsymbol{V}}.\end{aligned}$$

Then

$$\begin{aligned}J_{\boldsymbol{\Sigma}^{-1}\to\boldsymbol{U}} &= \left(\left|\frac{n-k}{a+p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2} + \boldsymbol{B}^{-1}\right|^{-1}\right)^{p} \\ &= \frac{1}{\left|\frac{n-k}{a+p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2} + \boldsymbol{B}^{-1}\right|^{p}}\end{aligned}$$

and

$$J_{\widehat{\boldsymbol{\Sigma}}\to\boldsymbol{V}} = \frac{|\boldsymbol{L}_2|^{(n-k-p-1)/2}\,|\boldsymbol{L}_1|^{(p+1-(n-k))/2}}{|(a+p-1)\,\boldsymbol{B}|^{(n-k)/2}}$$

by Lemma 4.32. Thus

$$J_{\boldsymbol{\Sigma}^{-1},\widehat{\boldsymbol{\Sigma}}\to\boldsymbol{U},\boldsymbol{V}} = \frac{|\boldsymbol{L}_2|^{(n-k-p-1)/2}\,|\boldsymbol{L}_1|^{(p+1-(n-k))/2}}{|(a+p-1)\,\boldsymbol{B}|^{(n-k)/2}\left|\frac{n-k}{a+p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2}+\boldsymbol{B}^{-1}\right|^{p}} \tag{4.73}$$

Applying standard change of variables with the Jacobian in Equation (4.73) gives

$$\begin{aligned} L(\boldsymbol{U},\boldsymbol{V}) = c_1\,|\boldsymbol{L}_1|^{(n-k-p-1)/2}\left|\boldsymbol{U}\left(\frac{n-k}{a-p+1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2}+\boldsymbol{B}^{-1}\right)^{-1}\right|^{(n-k+a-p-1)/2} \\ \cdot\exp\left(-\frac{1}{2}\operatorname{Tr}\boldsymbol{U}\right)\cdot\frac{|\boldsymbol{L}_2|^{(n-k-p-1)/2}\,|\boldsymbol{L}_1|^{(p+1-(n-k))/2}}{|(a-p-1)\,\boldsymbol{B}|^{(n-k)/2}\left|\frac{n-k}{a-p+1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2}+\boldsymbol{B}^{-1}\right|^{p}} \end{aligned}$$

This is equivalent to

$$\begin{aligned} L(\boldsymbol{U},\boldsymbol{V}) = c_2\frac{1}{\left|\frac{n-k}{a+p-1}\boldsymbol{B}^{-1/2}\boldsymbol{V}\boldsymbol{B}^{-1/2}+\boldsymbol{B}^{-1}\right|^{(n-k+a+p-1)/2}}\,|\boldsymbol{L}_2|^{(n-k-p-1)/2} \\ \cdot|\boldsymbol{U}|^{(n-k+a-p-1)/2}\exp\left(-\frac{1}{2}\operatorname{Tr}\boldsymbol{U}\right) \end{aligned}$$

where

$$c_2 = c_1\frac{1}{|(a+p-1)\,\boldsymbol{B}|^{(n-k)/2}}$$

is a constant. This can be further simplified to

$$\begin{aligned} L(\boldsymbol{U}, \boldsymbol{V}) &= c_2 \frac{1}{\left|\boldsymbol{B}^{-1/2}\left(\frac{n-k}{a+p-1}\boldsymbol{V} + \boldsymbol{I}_p\right)\boldsymbol{B}^{-1/2}\right|^{(n-k+a+p-1)/2}} |\boldsymbol{L}_2|^{(n-k-p-1)/2} \\ &\quad \cdot |\boldsymbol{U}|^{(n-k+a-p-1)/2} \exp\left(-\frac{1}{2}\operatorname{Tr}\boldsymbol{U}\right) \\ &= c_3 \frac{1}{\left|\frac{n-k}{a+p-1}\boldsymbol{V} + \boldsymbol{I}_p\right|^{(n-k+a+p-1)/2}} |\boldsymbol{L}_2|^{(n-k-p-1)/2} |\boldsymbol{U}|^{(n-k+a+p-1)/2} \exp\left(-\frac{1}{2}\operatorname{Tr}\boldsymbol{U}\right) \end{aligned} \tag{4.74}$$

where

$$c_3 = c_2 \frac{1}{\left|\boldsymbol{B}^{-1}\right|^{(n-k+a+p-1)/2}}$$

is a constant. Using the definitions of the singular multivariate F and Wishart distributions, Equation (4.74) separates into

$$L(\boldsymbol{U}, \boldsymbol{V}) \propto F_p^{n-k}\left(n-k, a+p-1; \tfrac{a+p-1}{n-k}\boldsymbol{I}_p\right)(\boldsymbol{V}) \cdot W_p\left(\boldsymbol{I}_p, n-k+a\right)(\boldsymbol{U}). \tag{4.75}$$

Thus $\boldsymbol{\Sigma}^{-1}\left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right)$ is distributed as $W_p\left(\boldsymbol{I}_p, n-k+a\right)$ and $(a+p-1)\,\boldsymbol{B}^{1/2}\widehat{\boldsymbol{\Sigma}}\boldsymbol{B}^{1/2}$ is distributed as $F_p^{n-k}\left(p, n-k, a+p-1; \frac{a+p-1}{n-k}\boldsymbol{I}_p\right)$. □

Using this result, the multivariate RVM may also be applied to GLMs when the sample covariance matrix is singular. The generalized variances under the null and alternative hypotheses are computed as in the case of RVM with a nonsingular covariance matrix. The hyperparameters a and $\boldsymbol{B}$ are estimated from $\widehat{\boldsymbol{\Sigma}}$ using a numerical maximization routine, with the singular multivariate F distribution being fitted. The hyperparameters are then used to adjust the numerator and denominator sums of squares in the likelihood ratio test. Values of the likelihood ratio test are then compared to cutoff values from the χ^2 distribution with the adjusted degrees of freedom to determine significance.

4.4 Parametric Error Models for Intensities

A third type of multivariate error model would be a parametric model for both the probe intensities and the variances. This approach is implemented in the mmgMOS algorithm, which is extensively reviewed in Section 1.8.2. The error model is constructed by first modeling the probe level intensities; in the mmgMOS algorithm, the gamma distribution is selected for this purpose. After fitting the model for the intensities, the variance of the intensities can be computed using standard formulae. The mmgMOS method is used as one of the comparison algorithms in this research project to assess the accuracy of parametric models based on probe intensities.

Chapter 5

Performance Assessment

5.1 Introduction

One of the greatest challenges in the development of algorithms for the analysis of microarray data is assessing the performance of the new method (Choe *et al.*, 2005). Comparisons of one algorithm to another without a proper reference data set are of little use. Unless the true state of gene expression between experimental conditions is known, the investigator cannot determine whether algorithms with larger number of genes declared significant have greater accuracy in detecting true positives or simply have a higher number of false positives. Early studies relied on confirmation of microarray results by use of independent laboratory techniques, such as Northern blotting, quantitative PCR, or ISH (Karsten *et al.*, 2004). However, this approach is very labor-intensive and time-consuming, so that only a small number of genes can be independently verified. Thus, it typically provides validation that the genes with the most significant test statistics represent true positives, but gives little information about the overall sensitivity and specificity of a given algorithm.

An alternative approach to assessing the accuracy of algorithms is to perform comparisons using spike-in datasets. The general methodology for construction of spike-in datasets is to introduce specific cDNA fragments into an experimental medium at prespecified concentrations. Because the concentration of the spiked probes in each condition is known, the number of truly differentially expressed genes is known and can be used for calculation of measures of sensitivity and specificity. Thus, application of statistical algorithms to spike-in datasets also represent

an independent means of assessing the performance of the statistical method. Such datasets are constructed by a considerable investment of time and resources, but these initial costs incurred by creating the dataset have an enormous return, since subsequent analyses can be conducted by any number of investigators with relatively little expense. Variations in the design of spike-in datasets allow the assessment of algorithms in different manners. For example, one design may be to spike all genes at a single concentration on a chip; alternatively, the concentration may vary among genes. The former represents a simpler design and assessment, while the latter permits comparisons under perhaps more realistic conditions, in which the expression differs among genes. The experimental medium may be a hybridization solution with or without background RNA present. Again, the former is a simpler design and assessment. The latter represents more realistic conditions, in which the differentially expressed genes must be separated from a large number of genes that have an unchanging level of expression, but it has the potential drawback that the hybridization properties of the background RNA may not be fully characterized. Finally, as with any experiment, stringent quality control measures are necessary in the construction of spike-in datasets to avoid erroneous results. In particular, cross-hybridization among probes must be addressed so that the list of genes expected to be declared differentially expressed is accurate.

5.2 Overview of Spike-in Studies

The first spike-in dataset was created by Affymetrix using the U95 GeneChip. This is a subset of the data used in the development and validation of the MAS5 statistical analysis algorithm. As detailed in the documentation accompanying this dataset, 14 experimental groups were constructed from 14 spiked-in human genes arranged in a Latin Square design (Tables 5.1 and 5.2). Group 1 contains 2 genes, group 12 is empty, and the remaining groups contain 1 gene. The concentrations of the 14 gene groups in the first experiment are 0, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, and 1024pM. Each subsequent experiment rotates the spike-in concentrations by one group, so that experiment 2 begins with 0.25pM and ends at 0pM, on up to experiment 14, which begins with 1024pM and ends with 512pM. Experiments 13 and 14 contain four technical

Group Number	Probeset IDs
1	37777_at
	407_at
2	684_at
3	1597_at
4	38734_at
5	39058_at
6	36311_at
7	36889_at
8	1024_at
9	36202_at
10	36085_at
11	40322_at
12	empty
13	1091_at
14	1708_at

Table 5.1: Probeset Groupings for the Affymetrix U95 Spike-In Study

replicates, while experiments 1 through 12 contain only a single hybridization. Two sets of experiments were performed, one set with a complex RNA background and one without. The complex background consists of RNA isolated from biological sources, which would contain a variety of RNA species and is intended to model the background hybridization present in most biological experiments. For the Affymetrix U95 experiments, mRNA isolates from human pancreas were selected. Each experiment contains 3 replicates except one (experiment C with complex background), which contains 2 replicates.

Affymetrix later produced a second spike-in dataset using the human U133 GeneChip. This dataset uses a common complex cRNA derived from a human HeLa cell line as background. Fourteen separate hybridizations were performed in which 42 transcripts were spiked into the hybridization cocktail using a Latin Square design (Tables 5.3 and 5.4). These spike-in genes contain 30 transcripts corresponding to cDNA clones isolated from a human lymphoblast cell line. The remaining spike-in genes consist of foreign and artificial clones expected to show little hybridization with human GeneChip probes, with 4 being bacterial sequences used as eukaryotic controls and 8 being artificially engineered sequences believed to be unique to the human

Experiment	Probeset Group													
Number	1	2	3	4	5	6	7	8	9	10	11	13	14
1	0	0.25	0.5	1	2	4	8	16	32	64	128	512	1024
2	0.25	0.5	1	2	4	8	16	32	64	128	256	1024	0
3	0.5	1	2	4	8	16	32	64	128	256	512	0	0.25
4	1	2	4	8	16	32	64	128	256	512	1024	0.25	0.5
5	2	4	8	16	32	64	128	256	512	1024	0	0.5	1
6	4	8	16	32	64	128	256	512	1024	0	0.25	1	2
7	8	16	32	64	128	256	512	1024	0	0.25	0.5	2	4
8	16	32	64	128	256	512	1024	0	0.25	0.5	1	4	8
9	32	64	128	256	512	1024	0	0.25	0.5	1	2	8	16
10	64	128	256	512	1024	0	0.25	0.5	1	2	4	16	32
11	128	256	512	1024	0	0.25	0.5	1	2	4	8	32	64
12	256	512	1024	0	0.25	0.5	1	2	4	8	16	64	128
13	512	1024	0	0.25	0.5	1	2	4	8	16	32	128	256
14	1024	0	0.25	0.5	1	2	4	8	16	32	64	256	512

Table 5.2: Concentration Data for the Affymetrix U95 Spike-In Study. All concentrations are given in pM.

Group Number	Probeset IDs	Group Number	Probeset IDs
1	203508_at	2	204205_at
	204563_at		204959_at
	204513__s_at		207655_s_at
3	204836_at	4	207777_s_at
	205291_at		204912_at_at
	209795_at		205569_at
5	207160_at	6	209606_at
	205692_s_at		205267_at
	212827_at		204417_at
7	205398_s_at	8	206060_s_at
	209734_at		205790_at
	209354_at		200665_s_at
9	207641_at	10	203471_s_at
	207540_s_at		204951_at
	204430_s_at		207968_s_at
11	AFFX-r2-TagA_at	12	AFFX-r2-TagD_at
	AFFX-r2-TagB_at		AFFX-r2-TagE_at
	AFFX-r2-TagC_at		AFFX-r2-TagF_at
13	AFFX-r2-TagG_at	14	AFFX-LysX-3_at
	AFFX-r2-TagH_at		AFFX-PheX-3_at
	AFFX-DapX-3_at		AFFX-ThrX-3_at

Table 5.3: Probeset Groupings for the Affymetrix U133 Spike-In Study

genome. Other improvements of this dataset over the U95 dataset are a wider spread of RNA concentrations and smaller (18 micron) chip features scanned using improved technology (the Affymetrix GeneChip Scanner 3000).

GeneLogic has also created a spike-in dataset consisting of three conditions. The Dilution and AML Latin Square data were previously described in the performance assessment of the S-Score in Section 2.3. A second Latin Square dataset, called the Tonsil Latin Square dataset, was produced in a manner similar to that of the AML Latin Square. The former differs from the latter in that complex cRNA derived from a tonsil tissue sample was used for background hybridization and in the arrangement of the spike-in concentrations in the Latin Square design (Table 5.5). Each experiment contains 3 technical replicates.

Choe *et al.* (2005) have provided an even more ambitious spike-in dataset, called the Golden

Experiment Number	Probeset Group													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0	0.125	0.25	0.5	1	2	4	8	16	32	64	128	256	512
2	0.125	0.25	0.5	1	2	4	8	16	32	64	128	256	512	0
3	0.25	0.5	1	2	4	8	16	32	64	128	256	512	0	0.125
4	0.5	1	2	4	8	16	32	64	128	256	512	0	0.125	0.25
5	1	2	4	8	16	32	64	128	256	512	0	0.125	0.25	0.5
6	2	4	8	16	32	64	128	256	512	0	0.125	0.25	0.5	1
7	4	8	16	32	64	128	256	512	0	0.125	0.25	0.5	1	2
8	8	16	32	64	128	256	512	0	0.125	0.25	0.5	1	2	4
9	16	32	64	128	256	512	0	0.125	0.25	0.5	1	2	4	8
10	32	64	128	256	512	0	0.125	0.25	0.5	1	2	4	8	16
11	64	128	256	512	0	0.125	0.25	0.5	1	2	4	8	16	32
12	128	256	512	0	0.125	0.25	0.5	1	2	4	8	16	32	64
13	256	512	0	0.125	0.25	0.5	1	2	4	8	16	32	64	128
14	512	0	0.125	0.25	0.5	1	2	4	8	16	32	64	128	256

Table 5.4: Concentration Data for the Affymetrix U133 Spike-In Study. All concentrations are given in pM.

	Transcript									
Experiment number	BioB-5_at	BioB-M_at	BioB-3_at	BioC-5_at	BioC-3_at	BioDn-3_at	DapX-5_at	DapX-M_at	DapX-3_at	CreX-5_at	CreX-3_at
1	0.5	3	2	12.5	25	5	1	0.75	50	1.5	75
2	0.75	5	3	25	50	12.5	1.5	1	75	2	100
3	1	12.5	5	50	75	25	2	1.5	100	3	0.5
4	1.5	25	12.5	75	100	50	3	2	0.5	5	0.75
5	2	50	25	100	0.5	75	5	3	0.75	12.5	1
6	3	75	50	0.5	0.75	100	12.5	5	1	25	1.5
7	5	100	75	0.75	1	0.5	25	12.5	1.5	50	2
8	12.5	0.5	100	1	1.5	0.75	50	25	2	75	3
9	25	0.75	0.5	1.5	2	1	75	50	3	100	5
10	50	1	0.75	2	3	1.5	100	75	5	0.5	12.5
11	75	1.5	1	3	5	2	0.5	100	12.5	0.75	25
12	100	2	1.5	5	12.5	3	0.75	0.5	25	1	50

Table 5.5: Concentration Data for the GeneLogic Tonsil Latin Square Dataset. All concentrations are in pM.

Spike dataset. This experiment used the Affymetrix *Drosophilia* GeneChip, which has 14010 probesets. The samples for this experiment are divided into constant (C) and spike (S) conditions, each with 3 technical replicates. The hybridization cocktail for the dataset consisted of 3860 cRNAs of known sequence spiked in at specific concentrations. The concentrations of 1309 cRNAs differ between the C and the S conditions. The fold changes range from 1.2 to 4, with the condition S arrays always having the higher concentration (Table 5.6). The remaining 2551 sequences, having the same concentration in the C and S conditions, represent a well-defined background population. A total of 3866 probesets should be detected as being expressed, while 1331 probesets should be identified as differentially expressed between the two conditions. (These figures differ slightly from the number of spike-ins because some sequences match more than one probe, while a few sequences do not match with any probes.) With approximately 10% of the probesets differing between the C and S conditions, this dataset allows the evaluation of algorithms in a setting that more closely resembles the typical gene expression study.

5.3 Methods for Comparisons

5.3.1 Data

Data for the Affymetrix U95 and U133 datasets were downloaded in ZIP archive format from the Affymetrix website (http://www.affymetrix.com/support/technical/sample_data/datasets.affx). The Golden Spike dataset was downloaded as a ZIP archive from the corresponding author's website (http://www.ccr.buffalo.edu/halfon/spike). A CD-ROM containing the GeneLogic Dilution and Latin Square datasets in self-extracting archives was obtained free of charge by request from the company (http://www.genelogic.com/newsroom/studies/studies.cfm). Each dataset consists of a series of *.CEL files, with one file for each chip hybridized. A listing of the filenames associated with each experiment is provided in Appendix 2.

Table 5.6: Concentration Data for the Choe *et al.* Golden Spike Dataset

Pool number	Number of clones	Number of assigned Affymetrix probe sets	Assigned fold change (S vs C)	Amount of RNA added to each C chip (ug)	Amount of RNA added to each S chip (ug)
1	87	84	1.2	0.47	0.56
2	141	143	2	0.43	0.85
3	85	83	1.5	0.35	0.52
4	180	185	2.5	0.73	1.82
5	90	89	1.2	0.29	0.35
6	88	96	3	0.65	1.94
7	186	188	3.5	0.76	2.67
8	90	95	1.5	0.44	0.67
9	180	190	4	0.78	3.11
10	183	191	1.7	0.48	0.81
13	391	385	1	0.37	0.37
14	369	355	1	1.23	1.23
15	394	404	1	0.40	0.40
16	452	453	1	0.57	0.57
17	419	434	1	0.44	0.44
18	372	407	1	0.31	0.31
19	163	191	1	0.27	0.27

5.3.2 Data Processing

Three methods - RMA, Logit-T, and mmgMOS - were selected for comparison to the proposed S-Score and multivariate RVM methods. Although RMA is a probeset-level method rather than a probe-level method, it is one of the most widely used summary methods and commonly used as a comparator for assessing the performance of new algorithms. Logit-t and multi-mgMOS are alternative probe-level algorithms, and their inclusion will assess the merits of the proposed methods relative to other probe-level methods. In comparing the five methods using the spike-in datasets, the respective *.CEL files were read into the R programming environment version 2.5.1 using the *ReadAffy* function in the *affy* package version 1.14.1. Both RMA (Section 1.5.5) and mmgMOS (Section 1.8.2) generate expression summary values, which are then compared with standard statistical tests. RMA expression summaries were computed using the *rma* function in the *affy* package. Expression summaries for mmgMOS were computed using the *mmgmos* function in the *puma* package version 1.2.0.

The S-Score (Section 4.2), multivariate RVM (Section 4.3), and Logit-t (Section 1.8.1) algorithms produce a test statistic for each probeset on a GeneChip, which is a direct measure of expression change. Multichip S-Scores were computed using the *SScore* function in version 1.8.0 of the *sscore* package. Values for the SF and SDT parameters were calculated using the *ComputeSFandSDT* function in the same package. The Pooled S-Score values were computed using a custom modification of the *SScore* function. For Logit-t, the July 2003 version of the C source code was obtained from the authors and compiled using the GNU C compiler *gcc* version 4.0.1. This executable was called from within R, using the system call function, to compute the Logit-t values. For the multivariate RVM method, probe intensity values were extracted directly from the *.CEL files using the *intensity* function in the *affy* package. Only the PM values within a probe pair were used in analyses. The PM intensities were $\log_2$ transformed and centered about zero as recommended by Chu *et al.* (2002). Mappings of probes to probesets were obtained using the *pmindex* function.

All computations were performed on a Macintosh Powerbook system with a G4 PowerPC

processor running Mac OS X 10.4.9. Source code for all programs are provided in Appendix 1.

5.3.3 Selection of Baseline

Since analyses were conducted in a pairwise fashion between conditions, it was necessary to specify a baseline condition to which all other conditions were compared. For the four Latin Square datasets, Experiment 1 was selected as a baseline for analyses. For the Dilution dataset, only Experiments 9 through 14 had sufficient chips to conduct analyses using all algorithms, and Experiment 9 was used as a baseline for comparisons. The Golden Spike dataset contains only two conditions, with the control (C) condition used as the baseline. For attaining optimal performance, comparisons using each algorithm should identify all spiked probesets as differentially expressed. Identification of fewer probesets among the spike-ins would be false negative findings, while identification of probesets in addition to these would be false positive findings. Therefore, using this information, sensitivity and specificity of comparisons made with each algorithm can be estimated. Based on the concentration data from the various datasets, a number of false negatives was to be expected. For some probesets, the relative change between the two conditions, expressed by the fold change, was too low to be detected despite high concentrations of RNA. For other probesets, the absolute amount of RNA may be too low to generate sufficient signal for detection, despite a high fold change between conditions. Also, given the large number of hypotheses being tested simultaneously, a number of false positives would also be expected.

5.3.4 Quality Assessment and Data Integrity Checks

Prior to analysis, a quality assessment was performed on each chip. Because of the nature of the spike-in experiments, many tests for quality control, such as RNA degradation, could not performed. The primary quality control measures were assessment of linearity and lack of fit, which could be performed on a subset of the data. For the GeneLogic Dilution dataset, all spike-ins on each chip have the same concentration, so that linear effects of concentration could not be examined. However, the intensities of all probe sets at a fixed concentration level should be

similar under the assumption of linearity. Quantile-quantile plots of the MAS5 intensity values were used to examine the assumption that the intensities were from a single distribution with a common mean. For the Affymetrix U95 and GeneLogic Latin Square datasets, the spike-ins have differing concentrations, and a linear increase in signal intensity with increasing concentration would be expected on each chip. Plots of probeset concentration versus the MAS5 intensity value were generated for each chip. Visual inspection of linearity within a chip was supplemented with calculation of the R^2 value of the linear regression equation. Assessment of lack of fit could not be performed as there were not multiple probes at the same concentration on each chip.

For the Affymetrix U133 Latin Square and the Golden Spike dataset, several groups of spike-ins were present at differing concentrations, and each concentration level contained multiple probesets. Assessment of linearity was performed by visually inspecting plots and examining linear regression results as for the Affymetrix U95 and GeneLogic Latin Square datasets. Lack of fit statistics were computed for probesets at the same concentration on each chip to determine if significant differences existed. For the Golden Spike dataset, data integrity was also examined by comparing the mapping of probesets to concentration values by two different methods. The first method was the direct mapping of probesets to concentration available on the corresponding author's website. The second method was an indirect mapping of probesets to pool numbers, followed by a mapping of pool numbers to concentrations, using supplementary data from the original manuscript. A Perl script was written to compare the results of the indirect mapping to that of the direct mapping to determine if any discrepancies existed. In addition, the number of probesets assigned to each pool was checked against the values in the original manuscript for accuracy.

5.3.5 Statistical Analysis

For each of the five algorithms, statistical tests were conducted between the baseline condition and each of the remaining conditions in a dataset in a pairwise fashion. All replicates for each condition were included in the analysis. Expression summary values produced by RMA and mmgMOS

were compared using the functions in the *multtest* package version 1.14.0 from Bioconductor. A two-sample t-test with unequal variances was performed between each pair of conditions using the *mt.teststat* function. The resulting raw *p*-values were used for subsequent analyses; adjustment of the *p*-values by controlling FDR were not used so that results reflect the performance of the expression algorithm rather than the algorithm for FDR control.

The S-Score and Logit-t algorithms both produce test statistics that are easily converted into *p*-values without adjustment. The S-Score values, representing standard deviations from a mean of zero, were converted to *p*-values using the standard normal CDF

$$p\text{-value} = 2\left(1 - \mathbf{\Phi}\left(|S_s|\right)\right)$$

where S is the S-Score value for the probeset s. The Logit-t is converted to *p*-values using the CDF of the t distribution, with the degrees of freedom equal to the total number of arrays minus 2, as recommended by the authors (Lemon *et al.*, 2003).

For multivariate RVM, a mixed effects model was constructed for the transformed and centered probe-level intensities of each probeset s:

$$\boldsymbol{Y}_{cms} = \boldsymbol{\mu} + \boldsymbol{\beta}_c + \boldsymbol{b}_s + \boldsymbol{\varepsilon}_{cms} \quad m = 1, 2, \ldots, N_m; c = 1, 2, \ldots, N_c \tag{5.1}$$

For this model,

$$\boldsymbol{Y}_{cms} = \begin{bmatrix} y_1 & y_2 & y_3 & \ldots & y_{N_p} \end{bmatrix}'$$

is the $N_p \times 1$ vector of transformed intensities, $\boldsymbol{\mu} = \mu\boldsymbol{J}$ is the $N_p \times 1$ vector of mean intensities for probeset s, $\boldsymbol{\beta}_c = \beta_c\boldsymbol{J}$ is the $N_p \times 1$ vector of effects for the the cth treatment, and

$$\boldsymbol{b}_s = \begin{bmatrix} b_1 & b_2 & b_3 & \ldots & b_{N_p} \end{bmatrix}'$$

is the $N_p \times 1$ vector of effects for the pth probe of probeset s. These effects are assumed to be

fixed. The random error term $\boldsymbol{\varepsilon}_{cms}$ is assumed to have a multivariate normal distribution with an expectation of $\mathbf{0}$ and covariance matrix $\boldsymbol{\Sigma}$, or $\boldsymbol{\varepsilon}_{cms} \sim \text{MVN}(\mathbf{0}, \boldsymbol{\Sigma})$. Thus

$$\mathcal{E}(\boldsymbol{Y}_{cms}) = \boldsymbol{\mu} + \boldsymbol{\beta}_c + \boldsymbol{b}_s.$$

A compound symmetric covariance structure was chosen for modeling the relationship among the probes in a probeset, based on previous work by Archer *et al.* (2006), so that

$$\text{Var}(\boldsymbol{\varepsilon}_{cms}) = \begin{bmatrix} \sigma^2 & \sigma_c^2 & \sigma_c^2 & \dots & \sigma_c^2 \\ \sigma_c^2 & \sigma^2 & \sigma_c^2 & \dots & \sigma_c^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_c^2 & \sigma_c^2 & \sigma_c^2 & \dots & \sigma^2 \end{bmatrix} = \boldsymbol{\Sigma}$$

where σ^2 is the variance of and σ_c^2 is the covariance among the individual probes in the probeset.

Since the multivariate RVM method requires that the covariance matrix $\boldsymbol{\varepsilon}$ have the same dimension for all probesets in an analysis, which in turn requires the transformed intensity vector $\boldsymbol{Y}$ to have same number of probes within each probeset, a separate analysis was conducted for each probeset size. For the Affymetrix U133 GeneChip, probeset sizes range from 8 to 20. There are 21,765 probesets containing 11 probes, 482 probesets containing 16 probes, and 40 probesets containing 20 probes. These groups of probesets were used in the analysis. Although the sample size for the 40 probesets with 20 probes may be insufficient for accurate estimation of the hyperparameters, this group was retained as it contains several of the spike-in probes. The remaining groups of probesets contained 1 to 4 probesets in each group. The sample size for these groups was deemed too small to yield meaningful results, and these groups were excluded from the analysis. It is expected that the typical RVM analysis will only include those groups of probesets that are sufficiently large for adequate estimation of the hyperparameters, with smaller groups of probesets being excluded. The smaller groups contain probesets used for quality control that are unlikely to be of interest in most differential expression studies. A brief annotation of the probesets in these smaller groups is provided in Appendix C.

Model fitting was performed using the *gls* function from version 3.1-83 of the *nlme* package, available from the Comprehensive R Archive Network (CRAN; http://cran.r-project.org). Maximum likelihood, rather than restricted maximum likelihood, was used to permit likelihood ratio tests of the fixed treatment effects (see Pinheiro and Bates, 2000, p. 83). As the *gls* function does not allow for multivariate response variables, the model in Equation (5.1) was modified slightly by "stacking" the responses for each probeset s into a single $\left(N_c \cdot N_m \cdot N_p\right) \times 1$ vector and introducing additional indicator variables denoting membership in each array and treatment (see http://www.cmm.bris.ac.uk/learning-training/multilevel-m-software/reviewr.pdf for a general discussion of this issue in R). Using a reference cell model with the first level of treatment and probe effects as the reference level (which is the default in R), the fixed terms in Equation (5.1) may be combined as

$$\boldsymbol{\beta} = \begin{bmatrix} \mu & \beta_2 & \beta_3 & \ldots & \beta_{N_c} & b_2 & b_3 & \ldots & b_{N_p} \end{bmatrix}'.$$

Equation (5.1) then may formulated as the multivariate model

$$\boldsymbol{Y}_s = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}_s \tag{5.2}$$

where $\boldsymbol{X}$ is an $\left(N_p + N_c - 1\right) \times 1$ matrix of indicator variables for the fixed effects, and $\boldsymbol{Y}$ is the matrix of intensity values formed by concatenating the intensity vectors $\boldsymbol{Y}_{cms}$ as

$$\boldsymbol{Y}_s = \begin{bmatrix} \boldsymbol{Y}_{11s} & \boldsymbol{Y}_{21s} & \ldots & \boldsymbol{Y}_{c1s} & \ldots & \boldsymbol{Y}_{cms} \end{bmatrix}.$$

The random error matrix $\boldsymbol{\varepsilon}_s$ may be similarly formed as

$$\boldsymbol{\varepsilon}_s = \begin{bmatrix} \boldsymbol{\varepsilon}_{11s} & \boldsymbol{\varepsilon}_{21s} & \ldots & \boldsymbol{\varepsilon}_{c1s} & \ldots & \boldsymbol{\varepsilon}_{cms} \end{bmatrix}. \tag{5.3}$$

Then $\mathcal{E}(\boldsymbol{\varepsilon}_s) = \mathbf{0}$, where in this case the matrix $\mathbf{0}$ is an $N_p \times N_m N_c$ matrix.

The multivariate model in Equation (5.2) is converted to a "univariate" model using the vec

operator and Kronecker products, as described in Searle (1978) and Henderson and Searle (1979). The stacked "univariate" model is then

$$\begin{aligned}\text{vec}\left(\boldsymbol{Y}_s\right) &= \text{vec}\left(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}_s\right) \\ &= \text{vec}\left(\boldsymbol{X}\boldsymbol{\beta}\right) + \text{vec}\left(\boldsymbol{\varepsilon}_s\right).\end{aligned}$$

Applying Equation (4.7), this becomes

$$\text{vec}\left(\boldsymbol{Y}_s\right) = \left(\boldsymbol{I} \otimes \boldsymbol{X}\right) \text{vec}\left(\boldsymbol{\beta}\right) + \text{vec}\left(\boldsymbol{\varepsilon}_s\right). \tag{5.4}$$

Since $\mathcal{E}\left(\text{vec}\left(\boldsymbol{\varepsilon}_s\right)\right) = \text{vec}\left(\mathcal{E}\left(\boldsymbol{\varepsilon}_s\right)\right)$,

$$\mathcal{E}\left(\text{vec}\left(\boldsymbol{\varepsilon}_s\right)\right) = \mathbf{0}$$

and

$$\mathcal{E}\left(\text{vec}\left(\boldsymbol{Y}_s\right)\right) = \left(\boldsymbol{I} \otimes \boldsymbol{X}\right) \text{vec}\left(\boldsymbol{\beta}\right),$$

where $\mathbf{0}$ is an $N_p N_m N_c \times 1$ column vector. The term $\text{vec}\left(\boldsymbol{\varepsilon}_s\right)$ may be partitioned as

$$\text{vec}\left(\boldsymbol{\varepsilon}_s\right) = \begin{bmatrix} \boldsymbol{\varepsilon}_{11s} \\ \boldsymbol{\varepsilon}_{21s} \\ \boldsymbol{\varepsilon}_{31s} \\ \vdots \\ \boldsymbol{\varepsilon}_{c1s} \\ \vdots \\ \boldsymbol{\varepsilon}_{cms} \end{bmatrix}. \tag{5.5}$$

With the partitioning given in Equation (5.5), the formula for $\text{var}(\text{vec}(\boldsymbol{\varepsilon}_s))$ is

$$\begin{aligned}
\text{var}(\text{vec}(\boldsymbol{\varepsilon}_s)) &= \begin{bmatrix}
\text{var}(\boldsymbol{\varepsilon}_{11s}) & \text{cov}(\boldsymbol{\varepsilon}_{11s}, \boldsymbol{\varepsilon}_{21s}) & \text{cov}(\boldsymbol{\varepsilon}_{11s}, \boldsymbol{\varepsilon}_{31s}) & \dots & \text{cov}(\boldsymbol{\varepsilon}_{11s}, \boldsymbol{\varepsilon}_{cms}) \\
\text{cov}(\boldsymbol{\varepsilon}_{21s}, \boldsymbol{\varepsilon}_{11s}) & \text{var}(\boldsymbol{\varepsilon}_{21s}) & \text{cov}(\boldsymbol{\varepsilon}_{21s}, \boldsymbol{\varepsilon}_{31s}) & \dots & \text{cov}(\boldsymbol{\varepsilon}_{21s}, \boldsymbol{\varepsilon}_{cms}) \\
\text{cov}(\boldsymbol{\varepsilon}_{31s}, \boldsymbol{\varepsilon}_{11s}) & \text{cov}(\boldsymbol{\varepsilon}_{31s}, \boldsymbol{\varepsilon}_{21s}) & \text{var}(\boldsymbol{\varepsilon}_{31s}) & \dots & \text{cov}(\boldsymbol{\varepsilon}_{31s}, \boldsymbol{\varepsilon}_{cms}) \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\text{cov}(\boldsymbol{\varepsilon}_{cms}, \boldsymbol{\varepsilon}_{11s}) & \text{cov}(\boldsymbol{\varepsilon}_{cms}, \boldsymbol{\varepsilon}_{21s}) & \text{cov}(\boldsymbol{\varepsilon}_{cms}, \boldsymbol{\varepsilon}_{31s}) & \dots & \text{var}(\boldsymbol{\varepsilon}_{cms})
\end{bmatrix} \\
&= \begin{bmatrix}
\boldsymbol{\Sigma} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \boldsymbol{\Sigma} & \mathbf{0} & \dots & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma} & \dots & \mathbf{0} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \boldsymbol{\Sigma}
\end{bmatrix}, \qquad (5.6)
\end{aligned}$$

or, more compactly, $\text{var}(\text{vec}(\boldsymbol{\varepsilon}_s)) = \boldsymbol{\Sigma} \otimes \boldsymbol{I}_{mc} = \text{var}(\text{vec}(\boldsymbol{Y}_s))$. The variable $\boldsymbol{Y}_s$ is said to have a matrix variate normal distribution (Dawid, 1981), denoted in this case as $\boldsymbol{Y} \sim N_{N_{p,mc}}(\boldsymbol{X\beta}, \boldsymbol{\Sigma} \otimes \boldsymbol{I}_{mc})$.

After fitting the model in Equation (5.4) using the *nlme* package, the residuals from this analysis were then used to estimate the parameters a and $\boldsymbol{B}$ of the Wishart prior, as given in Equation (4.62). Parameter estimates were found using numerical optimization as implemented in the *optim* function in the *stats* package. The *optim* function minimizes a specified function using an implementation of the Nelder-Mead simplex algorithm (Nelder and Mead, 1965), which is relatively robust to discontinuities compared to the Newton-Raphson algorithm and does not require a gradient for the function being optimized. After obtaining estimates for a and $\boldsymbol{B}$, revised likelihood ratio test statistics were computed on a probeset-by-probeset basis using Equation (4.48). The p-values were obtained from the χ^2 distribution with 1 degree of freedom using -2 times the logarithm of the likelihood ratio test statistic.

The analyses differed between the GeneLogic Dilution dataset and the remaining datasets because of the differing nature of the experiments. For the GeneLogic Dilution dataset, all probesets on a chip were spiked in at the same concentration, so that the effects of concentration on the de-

tection of differential expression could only be examined across chips. The primary measure of performance was the sensitivity and specificity of each of the five algorithms. Higher values of sensitivity and specificity would indicate better performance of a particular method. To calculate these two quantities, it was necessary to establish cutoffs for declaring probesets differentially expressed. For multivariate RVM, RMA, and mmgMOS, a cutoff value of $p < 0.001$ was used, as suggested by Simon *et al.* (2002). A cutoff value of 3.29 was used for the absolute values of the S-Score, corresponding to greater than 3.29 standard deviations of change in intensity or $p < 0.001$. This is slightly higher than the previously recommended cutoff of 3, which corresponds to $p < 0.003$. For Logit-t, the *t*-test value corresponding to $p < 0.001$ was used as the cutoff value. This is slightly lower than the cutoff of $p < 0.01$ recommended by the authors (Lemon *et al.*, 2003). The use of these cutoff values for the S-Score and Logit-t is intended to provide uniformity across methods, so that differences in performance are not due to differences in cutoff values. Sensitivity and specificity were tabulated for each method based on the appropriate cutoff values. This was supplemented with plots of the S-scores and of multivariate RVM versus each of the three remaining algorithms to assess the comparative ability of each algorithm to clearly separate the spike-in clones from the remaining probe sets.

A different approach was used for the Latin Square and Golden Spike datasets, which contained varying concentrations of spike-in transcripts on each chip. Probe sets were rank ordered based on *p*-values obtained from each algorithm, using the *rank* function in R. Rankings from each algorithm were compared to the true underlying fold-change values of the spike-in clones. The true underlying fold-change ranks were determined using the concentration of the spike-in clones (Tables 2.2, 5.2, 5.4, 5.5, and 5.6) for the two conditions being compared. The proportion of spike-ins ranked less than or equal to the total number of spike-ins for the dataset was calculated, and the Cochran-Mantel-Hanzel test used to compare these proportions across all chips. This validation procedure is similar to the procedure for validation of the original S-Score using spike-in data (Kennedy *et al.*, 2006a).

5.4 Results

5.4.1 Quality Assessment

A subset of the quantile-quantile plots of the MAS5 intensity values for the GeneLogic Dilution dataset are depicted in Figure 5.1, with the full set of 26 plots in Appendix B. These plots generally show the assumption of linearity is reasonable; that is, the intensities of the spike-in probes are from a single underlying distribution. Two chips have a single probe falling outside of the 95% confidence bands of the quantile-quantile plot. With only two outliers among 26 hybridizations of 10 probes each, the quality of the Dilution dataset was deemed adequate, and this dataset was used in subsequent analyses.

Subsets of the linearity plots of the MAS5 intensity values for the Genelogic AML and Tonsil Latin Square datasets are shown in Figures 5.2 and 5.3, respectively, with the full set of plots in Appendix B. As evident from these plots, there are some problems with the assumption of linearity for almost all chips in the two datasets; that is, the intensities of the spike-in probes do not increase linearly with increases in concentration, as would be expected. The visual results are confirmed by the linear regression of intensity on concentration. The R^2 values range from 0.01 to 0.98 (mean = 0.61, median = 0.74) for the AML dataset and from 0.31 to 0.99 (mean = 0.79, median = 0.92) for the Tonsil dataset. The large number of chips violating the assumption of linearity may indicate potential problems with the quality of these datasets, and both were excluded from further analyses.

Subsets of the linearity plots for the Affymetrix U95 and U133 Latin Square datasets are show in Figures 5.4 and 5.5, respectively, with the full set of plots in Appendix B. As evident from these plots, the assumption of linearity is reasonable for almost all chips in the two datasets. The results of the linear regression showed similar results. The R^2 values range from 0.33 to 0.91 (mean = 0.74, median = 0.77) for the U95 Latin Square dataset. The R^2 values for the U133 Latin Square dataset ranged from 0.79 to 0.96 (mean = 0.88, median = 0.88). The lack of fit test for the U133 dataset was significant, $p < 0.01$. This indicates that intensities at the same concentration level

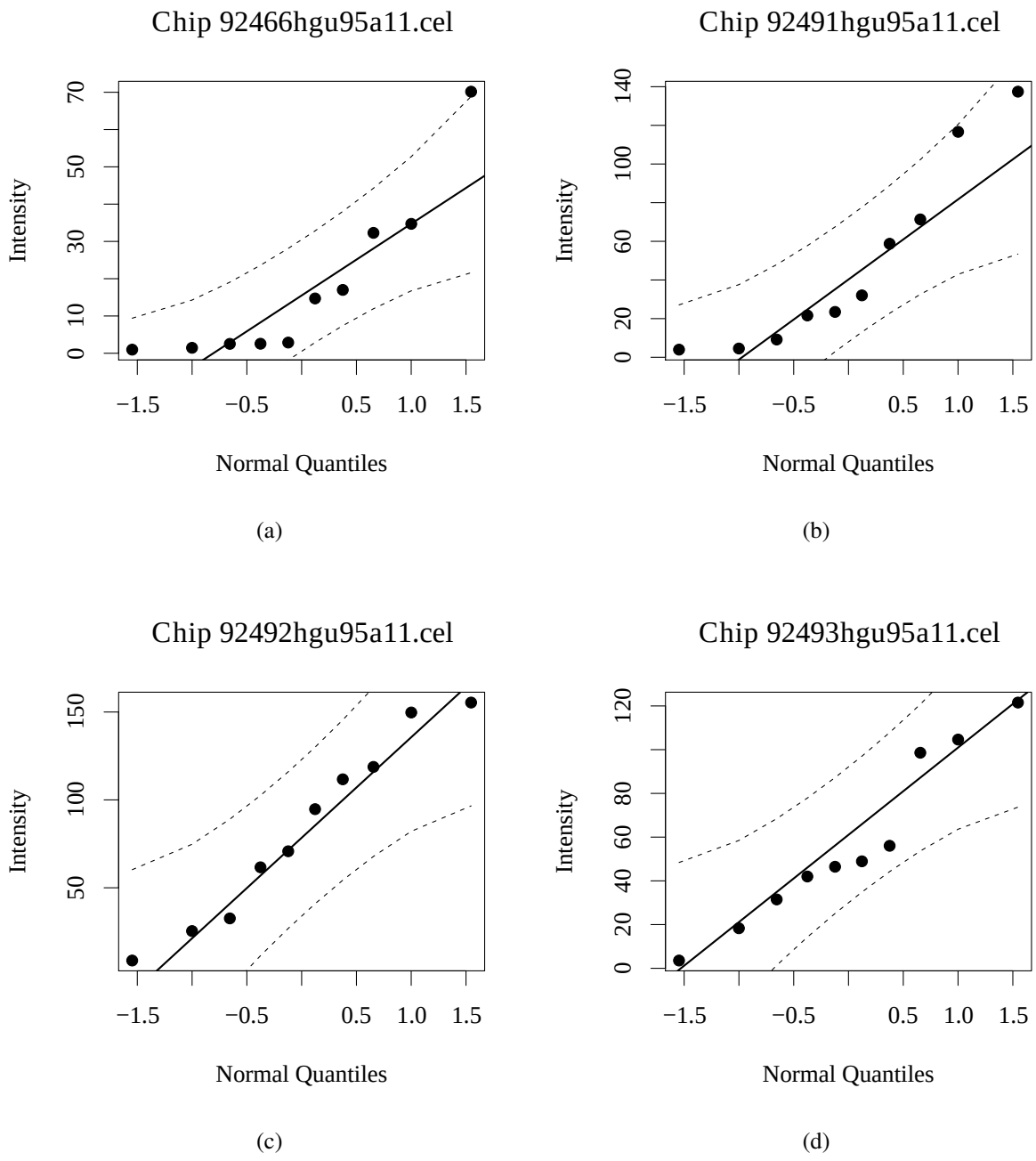

Figure 5.1: GeneLogic Dilution Quality. Plots of the computed MAS5 intensity values versus theoretical normal quantiles for a subset of chips. All intensity values are scaled to give a median intensity value of 100 for each chip.

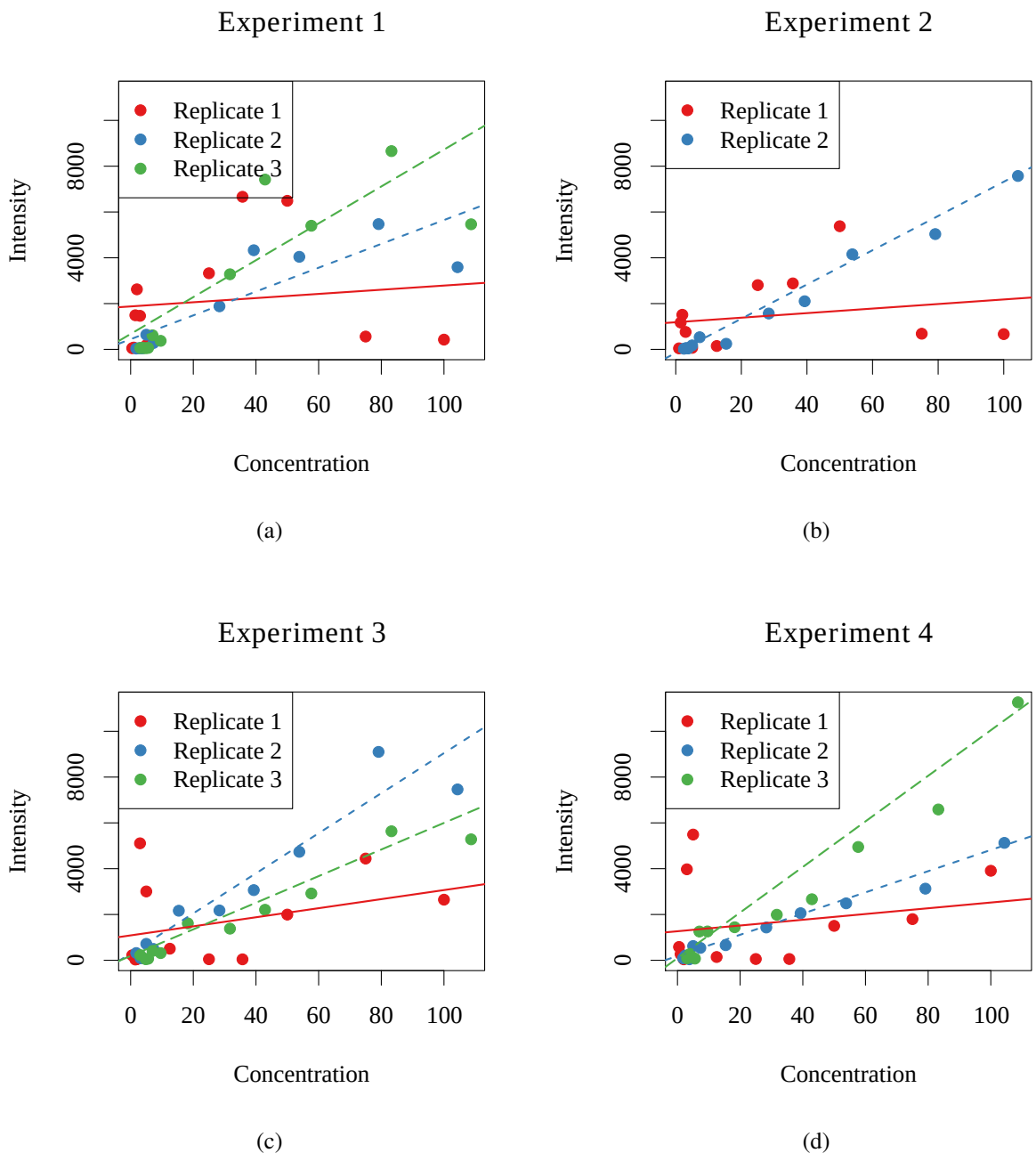

Figure 5.2: GeneLogic AML Latin Square Quality. Plots of the computed MAS5 intensity values versus concentration for a subset of chips. High-quality chips would be expected to show a linear increase in intensity as concentration increases. All intensity values are scaled to give a median intensity value of 100 for each chip.

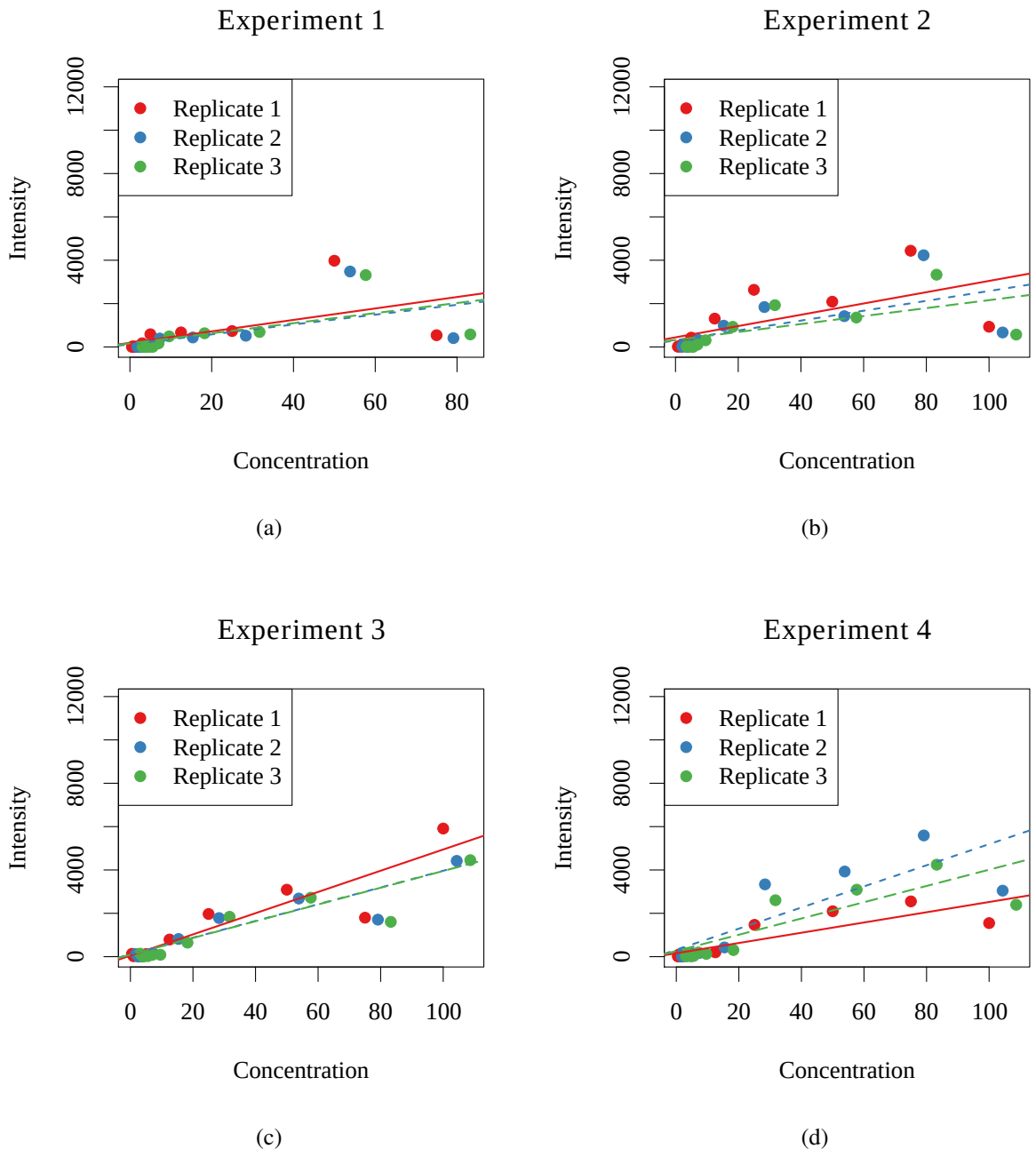

Figure 5.3: GeneLogic Tonsil Latin Square Quality. Plots of the computed MAS5 intensity values versus concentration for a subset of chips. High-quality chips would be expected to show a linear increase in intensity as concentration increases. All intensity values are scaled to give a median intensity value of 100 for each chip.

do not appear to come from a single underlying distribution, as would be expected. Although the lack of fit test may suggest potential problems with the U133 dataset, it was retained for further analyses based on the excellent results from the linearity plots. The U95 dataset was also retained, due to its inclusion in other benchmark studies (Irizarry *et al.*, 2006).

Quantile-quantile plots for the Golden Spike constant arrays and a linearity plot for the spike arrays are shown in Figure 5.6. As evident from these plots, there are significant deviations from the expected values, indicating that the intensities for the control chips are not from a single distribution. Similarly, the linearity plot shows that the probe intensities in the spike arrays do not increase linearly as a function of concentration, with the R^2 values ranging from 0.07 to 0.08. The lack of fit statistic for the spike arrays was highly significant, $p < 0.001$, suggesting that intensities at the same concentration level are not from a single underlying distribution as expected. Finally, results of the data integrity check using indirect mapping of probesets to pool numbers to concentrations is depicted in Table 5.7. The indirect mapping shows several discrepancies compared to the direct mapping in Table 5.6. One clone in the probeset file (SD01117) did not map to any pool assignment. There were also several pools (numbers 6, 13, and 14) for which the number of assigned clones and/or probesets differed between the direct and indirect mappings. These differences would affect the concentration value of only one probeset (152452_at), but do indicate potential problems with the quality of the Golden Spike dataset. Based on the results of the data integrity check, as well as the quantile-quantile and linearity plots, this dataset was excluded from further analyses.

5.4.2 Statistical Analysis

The results of the analysis of the GeneLogic Dilution dataset are shown in Tables 5.8 through 5.11. The statistical significance of the comparisons between different algorithms is given in Table 5.12. Both the multichip S-Score and Logit-T did well in detecting the spike-in probes for all but the lowest fold change. There were no significant differences in the performance of the two. RMA fared worse, failing to detect most of the spike-in probes except at the highest fold change. The

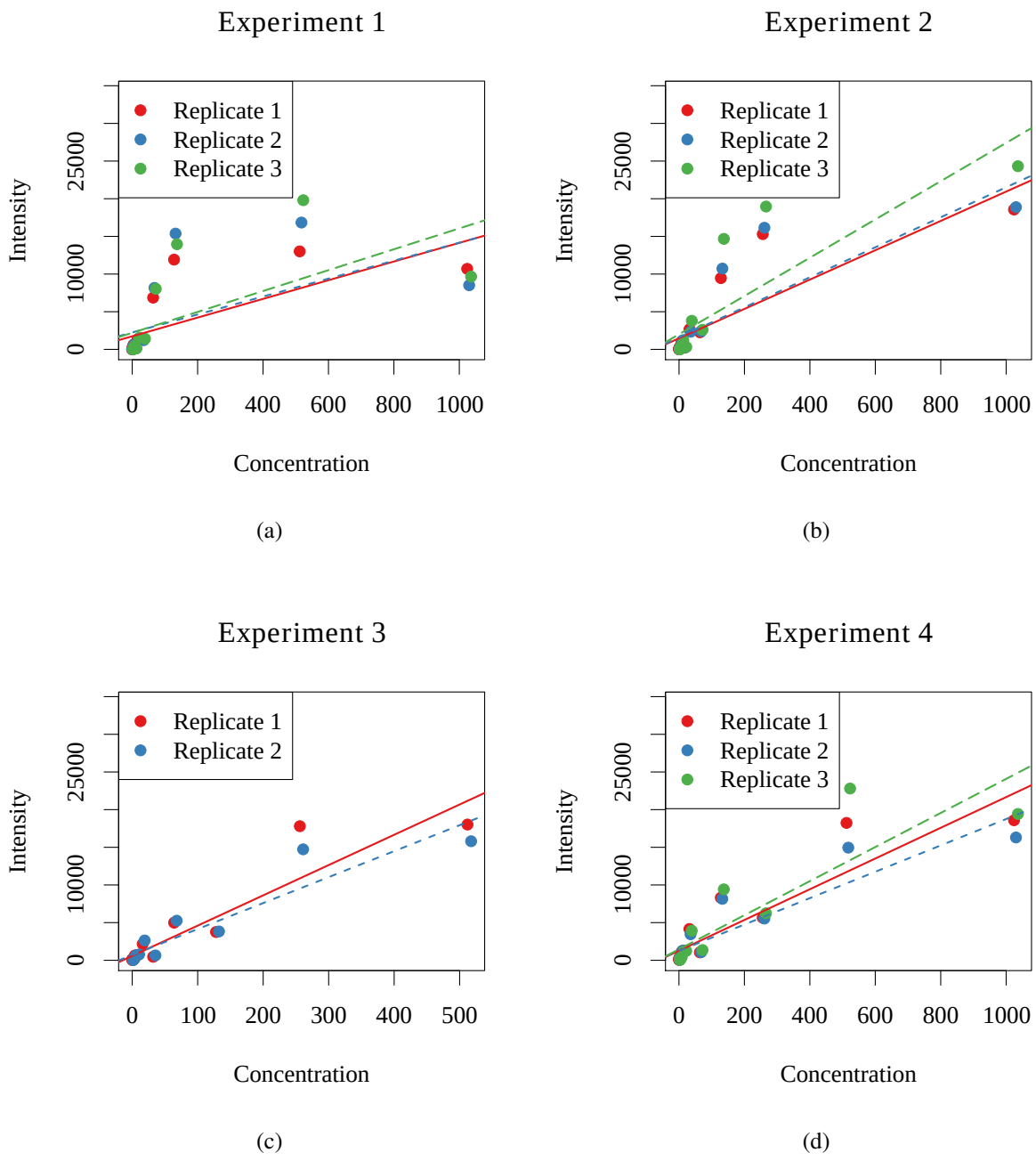

Figure 5.4: Affymetrix U95 Latin Square Quality. Plots of the computed MAS5 intensity values versus concentration for a subset of chips. High-quality chips would be expected to show a linear increase in intensity as concentration increases. All intensity values are scaled to give a median intensity value of 100 for each chip.

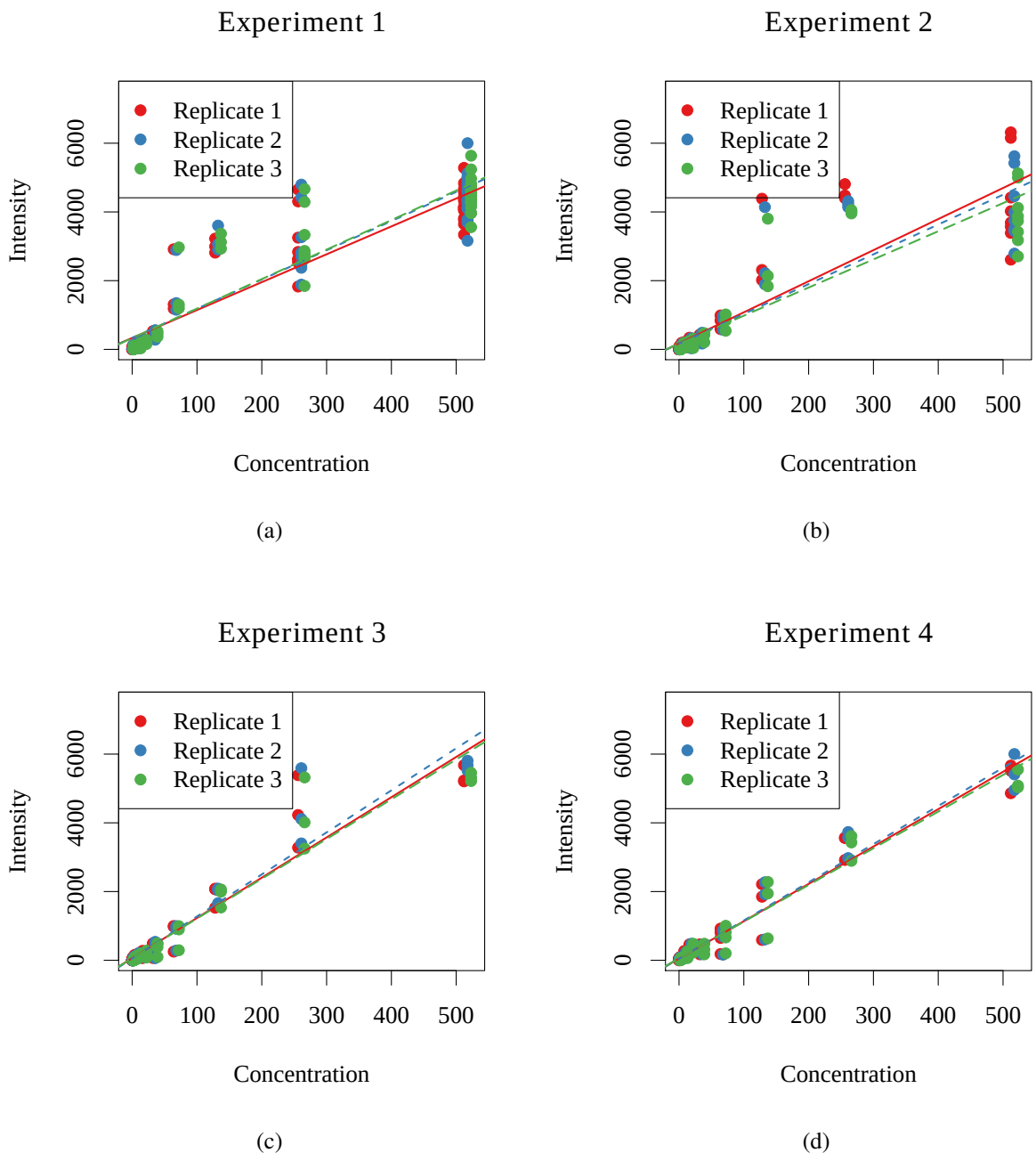

Figure 5.5: Affymetrix U133 Latin Square Quality. Plots of the computed MAS5 intensity values versus concentration for a subset of chips. High-quality chips would be expected to show a linear increase in intensity as concentration increases. All intensity values are scaled to give a median intensity value of 100 for each chip.

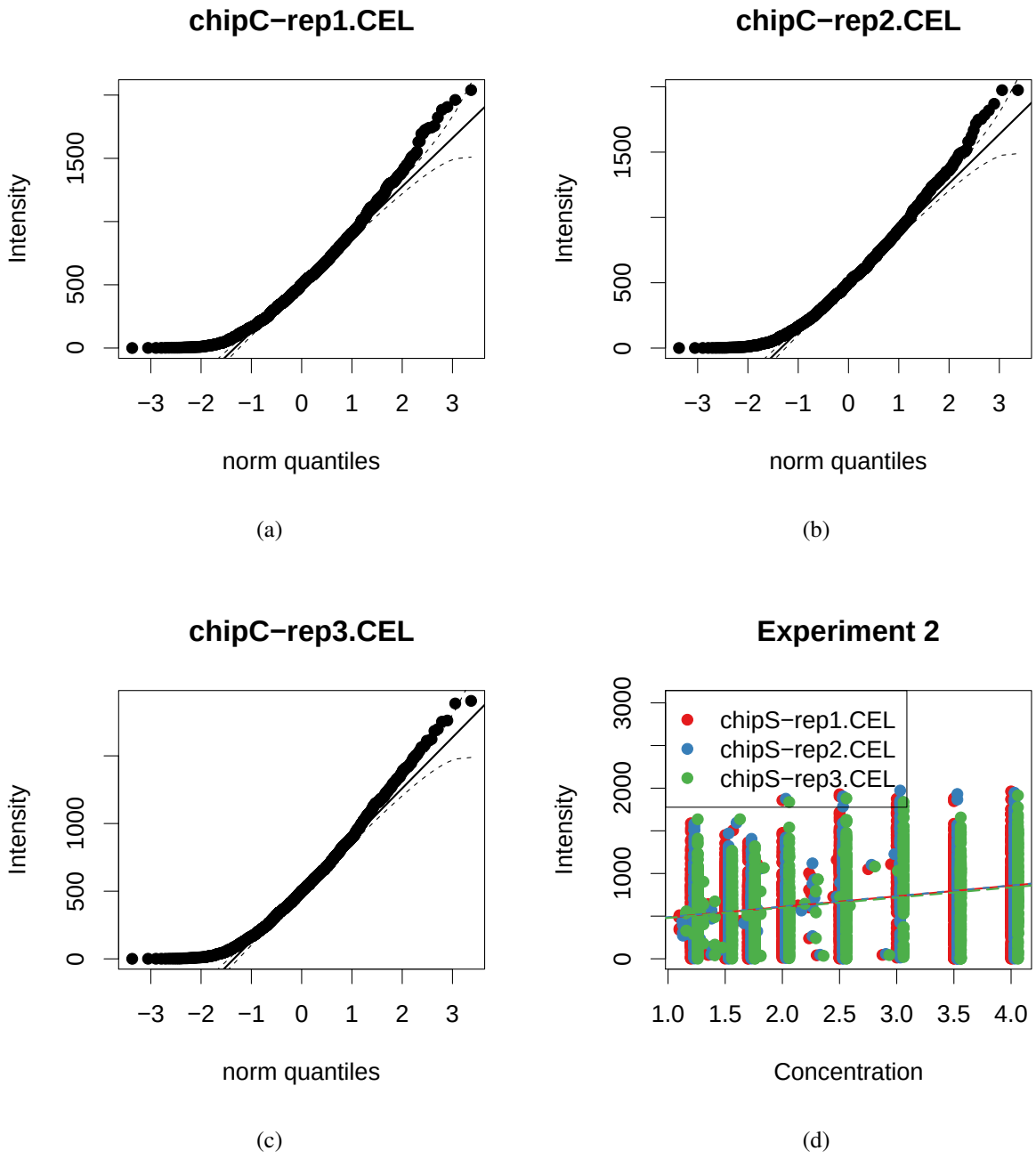

Figure 5.6: Choe Golden Spike Quality. All intensity values are scaled to give a median intensity value of 100 for each chip.

Pool number	Number of clones	Number of assigned Affymetrix probe sets
1	87	84
2	141	143
3	85	83
4	180	185
5	90	89
6	88	95
7	186	188
8	90	95
9	180	190
10	183	191
13	392	376
14	368	354
15	394	404
16	452	453
17	419	434
18	372	407
19	163	191

Table 5.7: Clone and Pool Assignments for the Choe *et al.* Golden Spike Dataset Using Indirect Mappings

results of the multichip S-Score and Logit-T analyses were significantly better than RMA. The mmgMOS, pooled S-Score, and RVM methods all performed quite poorly, often failing to detect any of the spike-ins. It should be noted that the analyses of Experiments 10, 13, and 14 using the RVM method failed to converge in the estimation of the matrix hyperparameter. Implications of nonconvergence on these results are discussed in the next chapter.

The results of the analysis for the Affymetrix U95 Latin Square dataset are shown in Table 5.13. The statistical significance of the comparisons between different algorithms is given in Table 5.14. The performance of the multichip S-Score was quite favorable, being significantly better than mmgMOS and RMA and comparable to Logit-T. For four experiments (numbers 1, 2, 3, and 11), the Logit-T detected a slightly higher number of spike-in probes than the multichip S-Score. For four other experiments (numbers 9, 16, 17, and 18) the multichip S-Score detected a slightly higher number of spike-in probes than the Logit-T. The pooled S-Score also showed significantly better results than mmgMOS and RMA. However, both the multichip S-Score and

Experiment	Logit-T	mmgMOS	Multichip S-Score	Pooled S-Score	RMA	RVM
10	0	0	5	0	0	0
11	9	2	10	0	1	0
12	10	0	10	0	0	0
13	10	0	9	0	3	0
14	10	0	10	0	9	2

Table 5.8: Number of True Positives in Analysis of the GeneLogic Dilution Dataset. All comparisons are made using Experiment 9 as the baseline chip. Values are the number true positives, i.e. the number of spike-in probes ranked in the top 10 according to the test statistic generated by each algorithm. The maximum number of spike-in probes that could be detected is 10. Note that analyses of Experiments 10, 13, and 14 using RVM had lack of convergence in estimating the matrix hyperparameter.

Experiment	Logit-T	mmgMOS	Multichip S-Score	Pooled S-Score	RMA	RVM
10	410	24	1	1	54	0
11	0	2	2	1	3	0
12	2	3	5	1	1	0
13	82	22	1	1	12	0
14	4	3	5	1	6	0

Table 5.9: Number of False Positives in Analysis of the GeneLogic Dilution Dataset. All comparisons are made using Experiment 9 as the baseline chip. Values are the number false positives, i.e. the number of non-spike-in probes ranked in the top 10 according to the test statistic generated by each algorithm. The maximum number of false positives is 12580 for the RVM method and 12616 for all other methods.

Experiment	Logit-T	mmgMOS	Multichip S-Score	Pooled S-Score	RMA	RVM
10	12206	12592	12615	12615	12562	12580
11	12616	12614	12614	12615	12613	12580
12	12614	12613	12611	12615	12615	12580
13	12534	12594	12615	12615	12604	12580
14	12612	12613	12611	12615	12610	12580

Table 5.10: Number of True Negatives in Analysis of the GeneLogic Dilution Dataset. All comparisons are made using Experiment 9 as the baseline chip. Values are the number true negatives, i.e. the number of non-spike-in probes not ranked in the top 10 according to the test statistic generated by each algorithm. The maximum number of true negatives is is 12580 for the RVM method and 12616 for all other methods.

Experiment	Logit-T	mmgMOS	Multichip S-Score	Pooled S-Score	RMA	RVM
10	10	10	5	10	10	10
11	1	8	0	10	9	10
12	0	10	0	10	10	10
13	0	10	1	10	7	10
14	0	10	0	10	1	8

Table 5.11: Number of False Negatives in Analysis of the GeneLogic Dilution Dataset. All comparisons are made using Experiment 9 as the baseline chip. Values are the number false negatives, i.e. the number of spike-in probes not ranked in the top 10 according to the test statistic generated by each algorithm. The maximum number of false negatives is 10.

	Algorithm					
Algorithm	Logit-T	mmgMOS	Multichip S-Score	Pooled S-Score	RMA	RVM
Multichip	2.690	67.955	—	74.116	43.959	67.955
S-Score	*0 .101*	*< 0.001*	—	*< 0.001*	*< 0.001*	*< 0.001*
Pooled	68.762	0.527	—	—	16.189	0.528
S-Score	*< 0.001*	0.468	—	—	*< 0.001*	0.468
RVM	62.339	0.260	—	—	11.243	—
	< 0.001	0.610	—	—	*< 0.001*	—

Table 5.12: Statistical Significance for True Positives the GeneLogic Dilution Analysis. The first row of each pair is the Cochran-Mantal-Haenzel test statistic, and the second row is the p-value obtained using the χ^2_1 distribution to determine significance.

Experiment	Logit-T	mmgMOS	Multichip S-Score	Pooled S-Score	RMA	RVM
2	9	2	8	7	4	0
3	10	0	9	7	3	0
4	12	6	11	10	11	0
5	12	10	12	9	10	0
6	12	10	12	12	10	0
7	13	11	13	12	10	4
8	13	9	13	12	12	1
9	13	9	13	12	10	0
10	12	8	13	9	11	0
11	12	8	12	12	11	1
12	11	7	10	7	6	0
13	10	5	10	10	7	0
13	10	6	10	8	6	0
13	10	4	10	8	7	0
13	10	5	10	8	8	0
14	5	2	9	6	1	0
14	7	2	9	7	1	0
14	7	2	9	6	4	2
14	8	4	8	8	5	1

Table 5.13: Statistical Analysis of the Affymetrix U95 Latin Square Dataset. All comparisons are made using Experiment 1 as the baseline chip, and Experiments 13 and 14 contain four technical replicates each. Values are the number of spike-in probes ranked in the top 14 according to the test statistic generated by each algorithm. The maximum number of spike-in probes that could be detected is 14. Note that all analyses conducted using RVM had lack of convergence in estimating the matrix hyperparameter.

Logit-T outperformed the pooled S-Score. Finally, the results obtained using the RVM method were significantly worse than the other five algorithms. For most of the experiments, the RVM method failed to detect any of the spike-in probes. However, it must also be noted that there was a lack of convergence in estimating the matrix hyperparameter for the RVM method with all 18 experiments as well. Implications of nonconvergence on these results are discussed in the next chapter.

The results of the analysis for the Affymetrix U133 Latin Square dataset are shown in Table 5.15. The statistical significance of the comparisons between the different algorithms is given in Table 5.16. As with the U95 Latin Square analysis, the multichip S-Score and Logit-T were

	Algorithm					
Algorithm	Logit-T	mmgMOS	Multichip S-Score	Pooled S-Score	RMA	RVM
Multichip	0.169	68.623	—	8.405	35.913	283.857
S-Score	*0.681*	*< 0.001*	—	*0.004*	*< 0.001*	*< 0.001*
Pooled	5.876	29.134	—	—	8.922	216.067
S-Score	*< 0.015*	*< 0.001*	—	—	*0.003*	*< 0.001*
RVM	274.490	115.355	—	—	161.051	—
	< 0.001	*< 0.001*	—	—	*< 0.001*	—

Table 5.14: Statistical Significance for the Affymetrix U95 Latin Square Analysis. The first row of each pair is the Cochran-Mantal-Haenzel test statistic, and the second row is the p-value obtained using the χ^2_1 distribution to determine significance.

comparable and outperformed the remaining algorithms. The pooled S-Score performed more poorly than the multichip S-Score and Logit-T. For this analysis, the pooled S-Score was also inferior to RMA, although it did outperform mmgMOS. The RVM method detected a number of spike-in probes and was superior to mmgMOS, but still fared poorly compared to the other four algorithms. However, in several instances, the estimation of the matrix hyperparameter for the RVM method failed to converge.

Experiment	Logit-T	mmgMOS	Multichip S-Score	Pooled S-Score	RMA	RVM
2	52	22	49	41	40	37
3	53	35	55	45	44	44
4	59	39	59	49	54	29
5	61	40	61	55	56	27
6	63	47	63	49	55	42
7	64	52	63	56	55	39
8	64	50	64	57	57	45
9	64	51	64	41	55	45
10	64	48	63	40	51	27
11	60	49	60	38	49	42
12	60	48	58	42	48	46
13	57	38	57	40	45	38
14	52	17	48	34	37	22

Table 5.15: Statistical Analysis of the Affymetrix U133 Latin Square Dataset. All comparisons are made using Experiment 1 as the baseline chip. Values are the number of spike-in probes ranked in the top 64 according to the test statistic generated by each algorithm. The maximum number of spike-in probes that could be detected is 64.

	Algorithm					
Algorithm	Logit-T	mmgMOS	Multichip S-Score	Pooled S-Score	RMA	RVM
Multichip	0.587	197.000	—	126.938	67.103	258.651
S-Score	*0.443*	*< 0.001*	—	*< 0.001*	*< 0.001*	*< 0.001*
Pooled	142.660	7.263	—	—	10.998	28.553
S-Score	*< 0.001*	*0.007*	—	—	*< 0.001*	*< 0.001*
RVM	278.170	7.289	—	—	74.329	—
	< 0.001	*0.007*	—	—	*< 0.001*	—

Table 5.16: Statistical Significance for the Affymetrix U133 Latin Square Analysis. The first row of each pair is the Cochran-Mantal-Haenzel test statistic, and the second row is the p-value obtained using the χ^2_1 distribution to determine significance.

Chapter 6

Discussion

6.1 Overview

This study represents the first analysis of probe-level algorithms using a comprehensive set of spike-in datasets. Although probe-level analysis has several potential advantages over probeset-level expression summary algorithms, claims of superiority for probe-level methods must be evaluated critically prior to acceptance. Previous studies of probe-level algorithms have several shortcomings that this study attempts to address. First, most studies have relied on datasets for which the true status of differential expression for each probeset is unknown, so that separation of true positives from false positives is difficult. Second, the studies that have utilized spike-in datasets were conducted prior to the development of some of the spike-in datasets available for this study. These newer datasets incorporate recent advances in microarray design that would be expected to lead to more accurate results, and would be more comparable to current microarray studies than analyses conducted a few years ago. Third, previous probe-level studies have not examined the quality of the datasets that are utilized, which may have a significant effect on the results obtained. Fourth, few comparisons between probe-level methods have been made, so that the relative advantages and disadvantages of different methods are not readily apparent.

6.2 Quality of Spike-In Datasets

The first set of significant results from this study concerns the quality of the available spike-in datasets, which was sufficiently poor to exclude many of the datasets from analyses. The nature of the spike-in experiments precludes the use of many standard tests for quality control for microarray data. For example, checks for the presence of ribosomal RNA degradation products are not useful, as these sequences would not be present in the spike-in studies even if significant RNA degradation had occurred. Similarly, assessment of the signal intensities for "housekeeping" genes would not be helpful as these sequences were not included in the spike-in sequences. Thus, while the available spike-in datasets are generally believed to be of good quality, this can be difficult to verify. Assessment of linearity is possible with spike-in datasets and provide indications of the quality of the experiments. The GeneLogic AML and Tonsil Latin Square datasets show significant deviations from the expected linear increase in intensity with increasing concentration, which is apparent from visual plots and from the low R^2 values. Similarly, the spike arrays of the Golden Spike dataset show significant departures from linearity, both visually and quantitatively. Such findings might be interpreted as evidence that the assumption of linearity may not hold for microarray data, particularly at low concentrations. Another interpretation would be that a linear increase in signal occurs for microarray data, but the slope of the line differs among probesets. If so, the increase in signal intensity for a given change in concentration that occurs with one probeset would not necessarily be the same as the increase in signal intensity that occurs with another probeset for the same change in concentration, and data for the individual probesets could not be combined. However, both the Affymetrix U95 and U133 Latin Square datasets show good results for linearity, which raises concerns that the results for the GeneLogic Latin Square and Golden Spike datasets represent poor quality in these experiments. The quantile-quantile plots for the constant arrays of the Golden Spike dataset would represent a check for linearity when all spikes are present at the same concentration. These plots show that the intensities are not from a single distribution, which is also a significant violation of the assumption of linearity. Quantile-quantile plots show that the assumption of linearity is reasonable for the GeneLogic Dilution dataset, al-

though there are a small number of probes that fall outside of the 95% confidence bands. Finally, it is concerning that the mappings of probesets to pool numbers could not be reproduced for the Golden Spike dataset using the published online data, as this may also be an indicator of poor quality for the experiment. Attempts to clarify these discrepancies with the authors were unsuccessful, so it cannot be established with certainty whether these findings represent errors in the original experiment or in the analyses from this study.

Taken collectively, the results of the quality assessment and data integrity checks for the spike-in datasets indicate that significant problems may exist with these datasets. This is of particular concern as these datasets are often used for establishing the accuracy of algorithms for detecting differential expression, which is in turn used for making judgments regarding the algorithms to pursue further. This study demonstrates that although spike-in experiments have shown improvements over time, current datasets may not be adequate for the purpose of algorithm development. Additional work is clearly needed regarding the design and creation of spike-in datasets to ensure that the validation of existing and future algorithms is performed properly.

6.3 The S-Score Algorithm

The second set of significant results from this study concerns the nature of probe-level analyses and how they are conducted. By extending the S-Score algorithm to multichip comparisons and the RVM algorithm to a multivariate model, this study advances both the theoretical underpinnings of probe-level methodology and the practical implementation of suitable analytic methods. For the S-Score algorithm, one of the chief concerns was that its development was based on empiric models that were intuitively appealing but not mathematically rigorous. The results of the current study provide initial results necessary for a more formal derivation of the S-Score method. The S-Score statistic closely resembles the sum in the Lindeberg-Feller generalization of the CLT. This would, in turn, predict that the S-Score statistic would approximately follow a standard normal distribution after appropriate scaling, which was observed by the original authors (Zhang *et al.*, 2002). The convergence of the S-Score statistic would occur regardless of the

underlying distribution of the signal intensities. This is a potential advantage over other methods that impose specific distributional assumptions, as the accuracy of the latter statistics may not hold if the assumptions are violated. If the underlying distribution of the signal intensities is approximately normal, the convergence of the S-Score statistic would be quite rapid. This would allow accurate results to be obtained even with small sample sizes, such as comparisons between two chips, which has also been observed (Kennedy *et al.*, 2006a). The CLT would also predict that the accuracy of the S-Score would increase as replicate chips were included in the statistic. This study extends the original R implementation of the S-Score algorithm to include replicate chips in the two conditions being compared and provides initial results on the performance of the extended version using spike-in datasets. Overall, the multichip S-Score did quite well, showing significantly better results than the standard probeset-level method RMA in the analysis of the Affymetrix Latin Square datasets. The multichip S-Score also did significantly better than the mmgMOS algorithm, which is a probe-level analysis method that assumes the signal intensities follow a gamma distribution. As the complex physical properties governing the binding of samples to probe sequences on oligonucleotide microarrays may not be fully appreciated at present, it would be quite understandable that a method that does not rely on underlying distributional assumptions may show superior performance to a method that does, as shown in this study.

However, a critical requirement for the applicability of the CLT to the S-Score statistic is that the denominator of the S-Score represent a consistent estimator of the variance of the signal intensities. This has not been established for the original S-Score or, by extension, to the multichip S-Score. Although the accuracy of the results obtained with these algorithms is encouraging, it does not constitute a proof. Thus, this study also sought to improve the variance estimate contained in the S-Score using an adaptation of the LPE algorithm, which was selected as its assumptions are similar to those of the S-Score. Surprisingly, use of the variance estimator based on the LPE algorithm led to a degradation in accuracy. The pooled S-Score did outperform RMA in the analysis of the Affymetrix U95 Latin Square dataset, but its performance was inferior to RMA for the Affymetrix U133 dataset. The pooled S-Score did not give comparable results to the multichip S-Score for any of the datasets examined. The poorer performance may be due

to limitations of the LPE method itself, particularly as applied to probe-level data. The variance estimate produced by the LPE method is the variance of a group of probes having a similar average intensity across chips. The upper tail of the distribution of signal intensities, which are often the signals of interest for differential gene expression studies, tend to be sparsely populated. This could lead to groups of probes having only a small number of members and in turn to a potentially inaccurate estimate of the true variance for the groups of probes, especially if outliers are present. The LPE algorithm attempts to correct the problem by requiring a minimum number of probes within each group. However, with the sparseness of the upper tails, this requirement could lead to probes with very different average intensities begin placed in the same group, which could inflate the variance estimate. Thus, the LPE method may produce inflated variance estimates for the probes with high intensity values. If such probes are likely to represent genes showing differential expression, the inaccuracies in the variance estimates would lead to excessively small S-Scores, making the detection of differential expression more difficult. Exploration of other methods for estimating the probe-level variances are warranted to determine if the inflated variance estimates can be eliminated and the performance of the pooled S-Score improved.

Taken together, these results provide a stronger theoretical background for the S-Score by demonstrating the similarity of the S-Score formula to the CLT. These results do remain limited by the fact that the variance estimate used in the calculation of the S-Score has not been established as a proper variance. Application of the multichip S-Score to spike-in dataset standards show excellent results, which is encouraging. The use of the LPE method to derive a more mathematically plausible variance estimate actually leads to poorer performance. This may be due to problems with the implementation of the LPE method or may reflect the accuracy of the S-Score variance estimator over other estimators.

6.4 The Random Variance Model

In contrast to the S-Score, the RVM method has a firm mathematical foundation by utilizing a general linear model framework for comparing conditions and assuming an inverse gamma

distribution for the error terms to achieve greater accuracy of the test statistics. However, it has only been developed as a probeset-level model, which may not have the same level of accuracy achieved by a probe-level model such as the S-Score. This study extends the RVM method to probe-level data by adopting a multivariate model with an inverse Wishart distribution for the error terms. The inverse Wishart is frequently viewed as a multivariate analogue of the gamma distribution. The inverse Wishart is also the prior of choice for multivariate Bayesian analysis, similar to the gamma for the univariate Bayesian model. This study provides two new proofs that give the distribution of the modified likelihood ratio test under the multivariate RVM assumptions and provide a method for estimating the hyperparameters used in the multivariate RVM method.

As originally formulated in this study, the multivariate RVM method requires that the sample variance-covariance matrix be a full rank, positive definite matrix. In the RVM method, this would occur with probability 1 if and only if the number of chips minus the number of conditions exceeds the number of probes in a probeset for a given Affymetrix chip type. Although studies using Affymetrix GeneChips are increasing in size, the large number of chips required by multivariate RVM with a nonsingular variance-covariance matrix remains unattainable for a large number of experiments. Accordingly, this study incorporates recent developments in singular multivariate distribution theory to derive the RVM method for the case of a singular sample variance-covariance matrix, although the matrix hyperparameter is still assumed to be of full rank. Two new proofs are given for the distribution of the modified likelihood ratio test under the singular multivariate RVM method as well as a method for estimating the hyperparameters. The relationship between the singular and nonsingular RVM formulae is also formally established, so that sets of computational routines specific for the singular and nonsingular cases can be developed for software implementations.

The assumption that the matrix hyperparameter $\mathbf{B}$ for the RVM method is of full rank, which implies that the scalar hyperparameter a is greater than or equal to the number of probes in a probeset, is required in the computation of the Jacobian of the transformations used in the multivariate F and beta distributions. The exact implications of this assumption on the RVM method are not clear at this time. The matrix hyperparameter $\mathbf{B}$, like the population covariance matrix $\mathbf{\Sigma}$,

is an unknown parameter; thus, its rank is unknown as well. Furthermore, although a consistent estimator of **B** may be obtained using the sample data in the singular case, the latter does not necessarily provide information about the rank of the former (Srivastava and von Rosen, 2002). Thus, it is difficult to determine if violations of the assumption that the matrix hyperparameter is of full rank occur. It should be noted that the current software implementation does restrict the hyperparameter a to be greater than or equal to the number of probes in a probeset, so that the pre-conditions on **B** are fulfilled. The estimates of the hyperparameters will still have the maximum likelihood given the sample data, subject to the constraint on a, and thus represent reasonable values for use in the multivariate RVM algorithm. However, as the optimization routines consistently find the value of a on this boundary, future studies deriving formulae for the RVM method when the matrix hyperparameter **B** is less than full rank seem warranted. Much of the theoretical research on singular multivariate distributions is quite recent, so that such applications of singular multivariate distributions must await further development of theory.

Given its theoretical rigor, and the favorable performance of the univariate RVM, the practical results of the multivariate RVM are disappointing. The multivariate RVM was consistently poorer than the multichip S-Score and Logit-T, which were the best performers. The multivariate RVM was also consistently poorer than the pooled S-Score and RVM, although the differences were less dramatic. It did outperform mmgMOS, though only on certain chips. Thus, the use of the multivariate RVM often led to the worst results of the six class comparison algorithms chosen. It is of course possible that these data indicate the RVM methodology is inappropriate for modeling the intensity data of Affymetrix GeneChips and making inferences from the constructed model. However, several other possible errors must be considered, many of which may be amenable to improvement in future research.

The first consideration must be whether the model for the probe-level intensities given in Equation (5.4) is correct. The residuals from this model are used in the construction of the likelihood ratio test in Equation (4.48) for hypothesis testing. The residuals are also used in estimating the hyperparameters through the RVM equation given in Equation (4.68) for the nonsingular case or Equation (4.75) for the singular case. Thus, misspecification of the intensity model

in Equation (5.4) may have tremendous impact on which genes are declared differentially or non-differentially expressed. The proposed intensity model, which incorporates fixed probe-level and treatment-level effects, appears reasonable from both a biological and statistical perspective. However, it might be argued that additional effects may be required for an adequate model, and that some effects might be represented better by random rather than fixed effects. As an example, there may be variation in the signal intensities due to the individual chip that is independent of both the treatment and probe effects. This would lead to the inclusion of a chip-level effect in the model of Equation (5.4). Such a chip-level effect would be appropriately modeled as a random effect, as the chips used in an experiment constitute a random sample of chips drawn from the population of chips that could have been selected. In the present work, consideration was given to the inclusion of a chip-level effect as well as a probe-level effect, but this approach was ultimately rejected due to limitations of the *nlme* package in fitting the model. As the probe effect is generally a greater source of variation than the chip effect in microarray experiments, the former was retained in constructing the model. The inclusion of other effects, such as percent GC content, may also be warranted but were not analyzed in the present work.

The hazards of underfitting and overfitting in general linear models are well-known in the case of fixed effects (see, for example, Myers, 1990, pp. 112-114). The effects of model misspecification on the RVM method have not been investigated to date, but are conjectured to lead to problems similar to those in the GLM. In underfitting, important predictor variables have been omitted from the model, and the variation due to these omitted variables is incorporated into the residual variance estimate. Depending on the nature of these predictors, the residuals could become significantly biased compared to their true values, although their variance remains unchanged. In the RVM method, the biased residuals would be expected to lead to inaccuracies in the likelihood ratio test, as it does in the GLM. The biased residuals may have additional adverse impact by affecting the estimate for the matrix hyperparameter $\boldsymbol{B}^{-1}$, which is also used in the calculation of the likelihood ratio test. In overfitting, predictor variables of marginal importance are included in the model. This leads to *variance inflation*, in which the estimates of the model variance are excessively large compared to their true values. In the RVM method, this variance

inflation would be expected to lead to wider confidence intervals for significance tests, difficulty in declaring genes to be differentially expressed, and larger numbers of false negatives.

A second consideration is the choice of the covariance structure used for the matrix hyperparameter $\boldsymbol{B}^{-1}$, which is in turn reflected in the structure of the covariance matrix $\boldsymbol{\Sigma}^{-1}$. Based on the work of Le *et al.* (1998), it appears that some structuring of the matrix hyperparameter is necessary to reduce the number of estimated variance components and avoid problems with identifiability. The compound symmetric structure, in which each of the different probes are assumed to have the same correlation, was chosen based on previous work of Archer *et al.* (2006) with pixel-level intensity data. In their work, the use of an first-order autoregressive structure, in which the correlation between different probes decreases with increasing distance, did not offer any significant advantages over the compound symmetric structure; other structures were not investigated. However, they note that this may reflect problems with distance metrics rather than the lack of advantage for more complex structures.

A third consideration is the choice of distribution for the variances in the RVM method; for the present work, the variance-covariance matrices are assumed to be distributed according to an inverted Wishart distribution. This is a logical choice as the inverted Wishart is the conjugate prior for the Wishart in Bayesian analysis. Furthermore, the inverted Wishart is the multivariate analogue of the inverted gamma distribution, which was used successfully in the development and validation of the univariate RVM. The inverted Wishart also offers computational convenience in the manipulation of the joint likelihood of the intensity measurements and their variance-covariance matrices, which is essential for the derivation of the RVM. Nevertheless, the form of the distribution for the variance-covariance matrices was chosen based on theoretical assumptions, and it is possible that other choices might result in a better fit. Further investigation of the effects of different distributional assumptions may be pursued in future work.

A fourth consideration is the model fitting method used for obtaining the estimates of the hyperparameters for the RVM method. In the present work, three parameters were varied to obtain empirical maximum likelihood estimates: the degrees of freedom a, the variance σ^2, and the covariance σ_c^2. The variance σ^2 constitutes the diagonal elements of the matrix hyperparameter

$\boldsymbol{B}^{-1}$, while the covariance σ_c^2 constitutes the off-diagonal elements. These three parameters were allowed to vary subject to the constraints $a \geq p$ and $\sigma^2, \sigma_c^2 > 0$. However, this is method of optimization is problematic, as many of the possible combinations of σ^2 and σ_c^2 under these constraints lead to a matrix $\boldsymbol{B}^{-1}$ that is not positive definite and thus not valid for the inverted Wishart distribution. The likelihood maximization routine does remove matrix hyperparameters that are not positive definite from consideration, but this creates multiple discontinuities in the likelihood function. The Nelder-Mead simplex algorithm used in the *optim* function is a derivative-free search method and relatively more robust to such discontinuities than other algorithms such as the Newton-Raphson. However, the number of discontinuities in the likelihood function still may lead to nonconvergence or convergence to a nonoptimal solution, even with the Nelder-Mead algorithm. Even if the optimization routine reports that convergence was achieved, the number of discontinuities may lead to convergence to a nonoptimal solution.

A more proper maximum likelihood estimation for $\boldsymbol{B}^{-1}$ would factor the matrix hyperparameter using the Cholesky or similar decomposition to produce a matrix where the individual elements may assume any values in a specified range with no discontinuities. These individual elements would then be allowed to vary over their range to locate a maximum likelihood estimate for $\boldsymbol{B}^{-1}$ using an optimization routine. Such an approach is more acceptable theoretically than the method used in the present work, but in practicality much more difficult to implement. The simple approach of allowing all of the individual elements within the decomposition matrix to vary ignores the fact that these elements are not independent due to the imposition of structure on the matrix $\boldsymbol{B}^{-1}$. Furthermore, the number of variables being optimized makes such an approach infeasible with current software. For example, using the Cholesky decomposition would produce a lower triangular matrix, so that $\frac{p(p+1)}{2}$ elements would need to be optimized individually. This would quickly overwhelm the capabilities of the *optim* function, which can optimize up to 20 variables simultaneously; other software packages have similar or greater restrictions. The more rigorous approach is to determine the dependencies among the individual elements of the decomposition matrix. Since, under compound symmetry, only two variables are needed for complete specification of $\boldsymbol{B}^{-1}$, only two variables (though in a different form) would be needed to specify

the elements of the decomposition matrix. The compound symmetry or similar structure would be well within the capabilities of current optimization software. However, the computation of the decomposition matrix based on the structure of $\boldsymbol{B}^{-1}$ is quite difficult to implement and has not been addressed to date in the literature. Thus, improvements in the optimization methodology, while promising, will require considerable time for the development of the relevant matrix formulae and construction of dedicated software.

A fifth and final consideration is the singularity of the covariance matrix used in the RVM for the spike-in datasets. This has the effect of restricting the subspace for the covariance matrix compared to the nonsingular case and reducing the amount of information contained in the sample. Thus, it is possible that the sample does not contain sufficient information for estimation of the population parameters, which may be accentuated in the singular case. If so, the performance of the RVM in the nonsingular case may be superior to other methods, even though the RVM in the singular case is not. Such a conjecture is interesting but cannot be properly evaluated with available spike-in datasets. All of the current spike-in datasets, as reviewed in this work, do not have an adequate number of chips to permit assessment of the nonsingular RVM. However, as research on standards for evaluation of microarray data analysis methods continues, it is likely that larger spike-in datasets will become available in the future. Verifying the performance of the nonsingular RVM should be a high priority for future research, as this may clarify the reasons for the singular RVM.

Taken collectively, these results demonstrate that the univariate RVM method can be successfully generalized to a multivariate one. Using multivariate distribution theory, new theorems are proven for the likelihood ratio test under the RVM assumptions, as well as the distribution of the modified sums of squares. Recent work on singular multivariate distributions are also incorporated to address small sample sizes, leading to theorems for a singular multivariate RVM. Application of the singular multivariate RVM to spike-in dataset standards, unfortunately, leads to poor results compared to other available class comparison methods. This may reflect problems with the implementation of the RVM method rather than deficiencies in the underlying model.

Chapter 7

Future Research

This study represents an initial contribution to the rapidly growing field of multivariate statistics and microarray data analysis. As such, part of the goal of this study is to develop lines of future research that will be pursued in later studies, which will be detailed in this section. Future work will focus on further refinements to the multivariate extensions of the S-Score and the RVM method, both of which show promise. The unexpected collateral findings for the quality of spike-in data will also be an important topic for future exploration.

7.1 Spike-In Datasets

Proper evaluation of algorithms for detecting differential expression in microarray experiments requires assessment using standardized datasets. The development of spike-in datasets represents a significant advancement in this assessment, as the differentially expressed genes are known and can be compared to the output of different algorithms. However, this study demonstrates that problems in quality occur with all of the presently available spike-in datasets. This is particularly true for the first spike-in datasets created. The quality has improved with later datasets, yet none could be considered entirely satisfactory. Additional work in the development of spike-in data will be essential for the development of microarray data analysis, as accurate comparisons among algorithms is not possible without a known standard for measuring differential expression.

An obvious avenue for future research is the creation of additional spike-in datasets. Such datasets would be expected to show improved quality due to refinements in chip design, hybridiza-

tion techniques, and analytical methods that have occurred since earlier spike-in experiments were performed. Newer datasets might also be larger than current ones, allowing the evaluation of multivariate analytic techniques without resorting to singular distributions. The creation of additional datasets is very feasible from a technical perspective, requiring that known RNA samples be titrated to specific concentrations, hybridized under strictly controlled conditions, and analyzed to produce the corresponding electronic data. Yet such experiments can be difficult to justify, as they represent a considerable investment of time and resources that may offer little immediate reward to the individuals and laboratories who undertake them. The promulgation of results such as those of this study, which demonstrate the deficiencies of current datasets, will likely be necessary to ensure that the creation of more spike-in datasets will be undertaken.

A promising alternative to spike-in datasets is the creation of RNA titration series for validating algorithms to detect differential gene expression. The development of titration series is newer than the development of spike-ins, but some datasets have begun to appear in the literature (Thompson *et al.*, 2005) and as part of the MicroArray Quality Control (MAQC) project (Shi *et al.*, 2006). Under this approach, two or more samples from different tissues are mixed together in a series of fixed ratios. In contrast to the spike-in datasets, the absolute levels of gene expression in the samples are not known, but the relative gene expression changes can still be determined based on the mixing ratios. Knowledge of these relative differences would be sufficient for assessing the performance of algorithms. Such experiments would be advantageous, as the number of differentially expressed genes would be larger than current spike-in datasets Such experiments may also be simpler to execute from a technical perspective, as the synthesis of spike-in clones is not necessary, but the requirements of time and resources may still make them difficult to justify.

Yet another alternative to spike-in datasets is the creation of RNA titration series *in silico* rather than *in vitro*. Under this approach, two or more samples from different tissues would still be combined in a series of fixed ratios, but mixing would be done at the level of the intensities from *.CEL files rather than at the level of extracted RNA solutions. An *in silico* titration would be quite promising, as the time and resource requirements for the creation of datasets would be

greatly reduced. It would also allow datasets to be expanded at a later date, which is not possible with current spike-in data. However, a greater technical investment would be needed for *in silico* than *in vitro* titration. The latter type of dataset would still need to be created to validate the results of the former, demonstrating that the two methods of titration lead to similar results. Such comparisons would necessitate new developments in equivalence testing, which has traditionally been applied to univariate data. Research in multivariate equivalence testing has only recently begun, and appropriate methods of equivalence testing for high-dimensional data have not been investigated to date.

A final avenue for research involving spike-in datasets is the development of quality control assessments. Research on this topic is growing rapidly, with a number of techniques being proposed to address this issue. However, as seen in the current study, spike-in datasets present unique challenges for quality control. The typical microarray experiment has a relatively large proportion of genes showing expression changes – approximately 10% – with the fold-changes varying over a wide range. Also, the full complement of cellular mRNA is used in the hybridization. In contrast, spike-in experiments may involve only a small number of genes, which are present with only a fixed number of fold-changes. Spike-in experiments may also lack specific RNA sequences, such as ribosomal RNA and messenger mRNA from "housekeeping" genes that are often used in quality assessments. The present study uses alternative methods, such as testing the assumption of linearity of hybridization signal with concentration, as quality measures. These analyses did demonstrate problems with the quality of existing spike-in datasets, but additional methods may be necessary. Assuming the continued development of spike-in datasets, these relatively simple methods may be adequate to detect gross quality defects, but not more subtle indicators of poor quality that may be present.

7.2 The S-Score Algorithm

The performance of the S-Score algorithm was excellent in the original studies utilizing two-chip comparisons. The present work extends the S-Score algorithm to incorporate multiple chips by

averaging the intensity and background measurements of chips by condition. This multichip S-Score compares quite favorably to other probe- and probeset-level models that are in current use. Such improvements are to be expected, given the similarity of the S-Score error formulation to the formulation of the Lindeberg-Feller generalization of the CLT. However, in order for the S-Score to fulfill the requirements of the CLT, its proxy error variance must be shown to be a consistent estimator of the true error variance. Demonstrating the consistency of the proxy error variance, or finding a substitute that performs comparably, constitutes the primary direction for future research on the S-Score.

The LPE algorithm represents a reasonable alternative to the proxy variance estimator of the S-Score algorithm. It is mathematically justifiable and in keeping with the intensity-based variance estimates used by the S-Score algorithm. The original and subsequent reports on the LPE algorithm also showed favorable results (Jain *et al.*, 2003; Park *et al.*, 2007). The adapted version of the LPE used in the current work, however, showed only modest ability to detect differentially expressed genes that was inferior to most of the other algorithms tested. Development of a version that more closely parallels the original LPE may result in improvements and will be explored in further work. For example, use of all pairwise comparisons to guarantee that the expected value of the intensity differences is 0 may offer benefits over the use of the mean intensity across all chips. Another line of research would be to explore other methods of creating intervals of adaptive widths in the high intensity region, so that more accurate estimates of the variance for these probes can be obtained. The tradeoff between potentially high variance estimates due to small numbers of probes versus potentially high variance estimates due to grouping disparate high intensity measurements requires further investigation to determine optimal methods.

A second area of research on the S-Score algorithm would be to study alternatives to the LPE method for obtaining a proxy variance estimate. Many of the intensity-based variance estimates assume a specific distribution for the variances, as in Sartor *et al.* (2006), Weng *et al.* (2006), or Hu and Wright (2007). Some distribution-free models do exist, but are probeset- rather than probe-level models. Eaves *et al.* (2002) used a weighted average of the variance of a probeset across chips and the mean of the variances for probesets with similar intensities. This includes aspects of

both the original LPE model and the adapted model used in the present work. Mansourian *et al.* (2004) obtain a robust estimate of the variance by dividing the probeset intensities into equal-sized bins, then computing the median of the variances for the probesets in each bin. Neuhäuser and Jöckel (2006) use a modified bootstrap as another nonparametric approach. For each gene, the probeset expression summaries are centered using the mean expression value of the gene across all chips. These centered expression summaries constitute the sample for obtaining the bootstrap estimate, with resampling performed at the gene level. Significance is determined with the *t* test comparing the original and bootstrap samples. Features of these approaches may be developed into probe-level implementations for the S-Score algorithm to determine if additional gains over the pooled S-Score can be obtained.

Finally, several minor improvements in the S-Score algorithm may be pursued in future research. Rewriting the code in a compiled language such as C would result in a significant gain in speed, particularly in the computation of the *SF* and *SDT* parameters that are required by the *SScore* function. This would also allow the open source Affymetrix routines for the computation of the *SF* and *SDT* to be incorporated into the code, assuring greater compatibility with the Affymetrix software. Creation of software routines that compute the S-Scores directly from the *.CEL files, rather than an object stored in memory, would reduce time and memory requirements in a manner similar to the *justRMA* function in the *affy* package. Such improvements would be included in future releases of the *sscore* package through the Bioconductor project.

7.3 The Random Variance Model

The RVM method is based on a sound theoretical foundation in multivariate statistical analysis. The imposition of a prior distribution for the covariance matrix has long been used in Bayesian analysis, and translates well to the frequentist approach. However, the practical results of the multivariate RVM method applied to available spike-in datasets is disappointing. Additional work is necessary to evaluate the RVM method and determine its usefulness in microarray data analysis.

One of the greatest concerns is that the nonsingular RVM method has not been evaluated,

due to limitations of current spike-in datasets. Thus it is difficult to ascertain the impact of the singular multivariate distribution, with its associated data reduction, on the results of the RVM method. The question of whether the nonsingular RVM method can achieve greater accuracy than the singular RVM cannot be answered without the development of larger spike-in or other standard datasets. Such datasets must have the number of chips minus the number of classes exceed the number of probes per probeset for the sample covariance matrix to be nonsingular; this would be 7 chips per class for a 2-class comparison involving the human U133 chip, but may require greater numbers for other chip types. If larger standardized datasets become available in the future, analysis using the nonsingular RVM method is straightforward, as existing code will need only slight modification to incorporate the nonsingular rather than singular multivariate distribution.

Another significant concern is the optimization routine for estimating the hyperparameters for the prior distribution, subject to the constraint that the matrix hyperparameter $\boldsymbol{B}$ be positive definite. The accuracy of this estimate affects the value of the likelihood ratio test, used for declaring genes differentially expressed, so that errors in the estimate can have a great impact on performance. Estimation of a positive definite covariance matrix is seen as a difficult problem without a standard solution (Schwallie, 1985). In many cases, the positive definite constraint is ignored in the estimation process. This is often unsatisfactory as it frequently leads to negative variance estimates that lack interpretability. Another option is to perform unconstrained estimation, then adjust the estimate so that the matrix is positive definite, using algorithms dedicated to this purpose (Hu and Olkin, 1991). This approach is usually unsatisfactory from a statistical perspective as the estimate is no longer a maximum likelihood estimate, and no longer possesses the desirable properties of the MLE. The option used in the current study is to assign an infinite value to the minus log likelihood (which corresponds to a negative infinite value for the likelihood) for any estimate that is not positive definite. This guarantees that a positive definite matrix will be selected for the maximum likelihood estimate, except in the improbable event that all calculated likelihoods are negative infinite. However, it is possible that assigning infinite values to the minus log likelihood may sufficiently distort the surface of the likelihood function that the optimization

routine cannot locate the correct solution. It is difficult to determine whether the optimization routine converges to an incorrect solution, indicating a need for further research to address the positive definite constraint.

The best option would be to transform the matrix hyperparameter and address the constraint within the transformation. Two possibilities for the transformation are the Cholesky decomposition and the matrix exponential function. For the Cholesky, the positive definite matrix $\boldsymbol{B}$ is written as

$$\boldsymbol{B} = \boldsymbol{B}^{*}\boldsymbol{B}^{*\prime},$$

where $\boldsymbol{B}^{*}$ is a lower triangular matrix. The individual elements of $\boldsymbol{B}^{*}$ are unconstrained, while still guaranteeing that $\boldsymbol{B}^{*}$ is positive definite. Thus the individual elements of $\boldsymbol{B}^{*}$ may be used as the parameters for the optimization routine to determine the estimate of $\boldsymbol{B}$. The matrix exponential $\boldsymbol{B} = \exp(\boldsymbol{B}^{+})$ may be calculated as

$$\exp(\boldsymbol{B}^{+}) = \sum_{i=0}^{\infty} \frac{(\boldsymbol{B}^{+})^{i}}{i!},$$

where $(\boldsymbol{B}^{+})^{0} = \boldsymbol{I}_{p}$ and

$$(\boldsymbol{B}^{+})^{i} = \overbrace{\boldsymbol{B}^{+} \cdot \boldsymbol{B}^{+} \cdot \ldots \cdot \boldsymbol{B}^{+}}^{i}.$$

This is the matrix analogue to the Taylor series expansion for the exponential of the scalar b, which is given by

$$\exp(b) = \sum_{i=0}^{\infty} \frac{b^{i}}{i!}.$$

If the matrix $\boldsymbol{B}^{+}$ is real and symmetric, then $\exp(\boldsymbol{B}^{+})$ is positive definite (Chiu *et al.*, 1996). Thus the individual elements of the lower triangle of $\boldsymbol{B}^{+}$ may be used as parameters for the optimization routine without constraint, with exponentiation of the corresponding symmetric matrix to com-

pute the estimate of $\boldsymbol{B}$. Both the Cholesky and the matrix exponential would appear suitable for the RVM method, though using a transformation would increase computational time and complexity. It is also unclear how the pattern of the hyperparameter $\boldsymbol{B}$ would be maintained when using a transformation.

A related area for further research would be the constraint that the matrix hyperparameter for the prior distribution be nonsingular. This constraint is a consequence of deriving the Jacobian of the transformation of $\widehat{\boldsymbol{\Sigma}}$ to $(a - p - 1)\,\boldsymbol{B}^{1/2}\widehat{\boldsymbol{\Sigma}}\boldsymbol{B}^{1/2}$, which is used to estimate the values of a and $\boldsymbol{B}$. Existing theorems for calculating this Jacobian require that the matrix $\boldsymbol{B}$ be nonsingular (Díaz-García and Gutiérrez Jáimez, 1997). Since, under singular RVM, the matrix $\boldsymbol{B}$ serves as a hyperparameter for the prior of the singular matrix $\widehat{\boldsymbol{\Sigma}}$, there is no reason to assume that $\boldsymbol{B}$ would be nonsingular; the constraint is merely a computational convenience. Extending current theorems regarding the above transformation would require considerable theoretical work to incorporate the generalized inverse of $\boldsymbol{B}$ into the Jacobian. The development of singular multivariate statistical theory is currently a rapidly growing topic of research, and the derivation of the Jacobian with a singular $\boldsymbol{B}$ matrix would be of great theoretical interest. If the accuracy of the singular RVM method improves with the application of the previously mentioned modifications, the derivation would be of practical importance as well.

Future work should also investigate the appropriateness of the large-sample χ^2 approximation for determining the significance of the likelihood ratio test statistic in the RVM method. Clearly, the sample sizes used in the spike-in dataset analysis raise the possibility that the approximation may not be adequate; a similar situation would arise in many microarray experiments. In the present study, the implementation of the RVM method does not allow the effects of the large-sample approximation to be disentangled from other potential sources of poor performance, but the lines of research detailed above should clarify this issue. If the approximation for the likelihood ratio test proves to be insufficient, potential remedial measures do exist. Ghosh and Sinha (1980) note that the likelihood ratio test remains valid if a prior distribution is imposed on the covariance matrix and subsequently integrated out of the likelihood. For the RVM method, this would transform the likelihood ratio test from the ratio of the maxima of two multivariate normal

distributions to the ratio of the maxima of two multivariate t distributions. This transformation may lead to a form of the likelihood ratio test for which an exact distribution is available. Alternatively, Dempster (1958, 1960) suggests abandoning the likelihood ratio test for high-dimensional data, instead choosing a suitable distance metric and then using the distribution of the data to develop an associated test of significance. Assuming multivariate normality, the proposed test statistic becomes the ratio of the trace of the hypothesis sums of squares to the trace of the error sums of squares, which approximately follows a χ^2 distribution. Under the RVM framework, Dempster's test would become

$$\frac{\mathrm{Tr}\left(\widehat{\boldsymbol{SS}} + \boldsymbol{B}^{-1}\right)}{\mathrm{Tr}\left(\widehat{\widehat{\boldsymbol{SS}}} - \widehat{\boldsymbol{SS}}\right)}.$$

Similar to the likelihood ratio test, Dempster's test would require that the term $\boldsymbol{\Sigma}^{-1}$ be factored from the term $\boldsymbol{\Sigma}^{-1}\left((n-k+a)\widehat{\boldsymbol{\Sigma}}\right)$ to derive the test statistic, which cannot be done. However, the general approach may still be useful if another suitable distance metric can be found in future studies.

The final major concern regarding the RVM method is the selection of the multivariate model for the intensities and their covariances. The current study uses a model for the intensities that incorporates a treatment effect and a probe effect to explain the observed differences between experimental groups. The covariance matrix for the intensity measurements is assumed to follow a compound symmetric structure. These choices are based on previous work on mixed effects models for intensities (Archer *et al.*, 2006) and current limitations of model-fitting software in the R programming environment. In future work, additional terms, such as a chip effect, would be incorporated into the mixed model, and formal statistical tests used to evaluate the degree of improvement in the model based on the new terms. Implementation of this will largely depend on further improvements in the R packages for fitting mixed effects models, or the porting of the RVM method to other statistical programming environments such as SAS IML or Stata. Future work should also explore alternative covariance structures, such as first-order autoregressive and

Toeplitz, as choices for the mixed model. Such alternatives did not offer any advantage over a compound symmetric structure in previous research Archer *et al.* (2006), but may be yield different results with the RVM method, particularly if coupled with new distance measures. The use of structured covariance matrices would appear to be necessary to reduce the number of parameters sufficiently to allow the maximum likelihood estimate to be computed.

There are also several minor improvements in the RVM method to be pursued in future research. Assuming that the investigations described above lead to improved accuracy of the RVM test statistic, the p-values that are obtained should be adjusted to control the false discovery rate. Such a modification would not be difficult to implement, but must account for the small sample sizes in estimating the null distribution for the FDR. Jain *et al.* (2005) have proposed the rank invariant resampling (RIR) method for estimating the null distribution that appears well suited to such situations.

The computational efficiency of the RVM method may also be improved rather easily. Considerable gains could be achieved by converting the code from the interpreted R language to the compiled C language. Additional gains might be realized by development of specialized optimization routines for estimating the hyperparameters under the RVM method, rather than using the general-purpose *nlme* package. Finally, again assuming that the above research produces a more accurate RVM test statistic, the method would be developed for release as an R package for the Bioconductor project.

Chapter 8

Conclusions

The purpose of this project is to expand the knowledge and use of probe-level analysis methods for Affymetrix GeneChip data by extending two promising models, the S-Score and the Random Variance Model. The original S-Score implements probe-level analysis but is limited as it performs only two-chip comparisons. This project extends S-Score algorithm by allowing comparisons among multiple chips. A simple averaging of the chip intensities and variances performs quite well on spike-in datasets, and appears reasonable based on the Central Limit Theorem. The difficulty with this approach is that the variance estimate is not well justified theoretically. The use of alternative variance models, such as the local pooled error algorithm, result in degraded performance, but other variance models are available and will be pursued in future work.

The RVM method is well justified theoretically, but the original formulation was limited to probeset-level data. This project extends the RVM method to a multivariate probe-level model, which is proven mathematically. The performance of the RVM implementation appears inferior to probeset-level and other probe-level methods, but may well be due to deficiencies in the associated software rather than the model itself. Several strategies for addressing this problem are proposed and will be explored in future work.

In summary, this project contributes to the growing research on probe-level analysis by advancing two previously existing models. Neither is optimal, as one still has theoretical issues and the other has practical issues that must be addressed. Still, this work shows the potential gains of probe-level analysis over traditional probeset-level methods, and shows that such work has a sound theoretical basis using recent developments in the theory of multivariate analysis. It is

hoped that this project will serve to stimulate further research into this rewarding topic.

Bibliography

Abadir, K. M. and Magnus, J. R. (2005). *Matrix algebra*, volume 1. Cambridge University Press, Cambridge.

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

Affymetrix (2002a). *GeneChip® Expression Analysis Technical Manual*. Affymetrix, Santa Clara, CA.

Affymetrix (2002b). *Statistical Algorithms Description Document*. Affymetrix, Santa Clara, CA.

Affymetrix (2004). *Affymetrix® GeneChip® Operating Software User's Guide*. Affymetrix, Santa Clara, CA.

Alberts, R., Terpstra, P., Hardonk, M., Bystrykh, L. V., de Haan, G., Breitling, R., Nap, J. P., and Jansen, R. C. (2007). A verification protocol for the probe sequences of Affymetrix genome arrays reveals high probe accuracy for studies in mouse, human and rat. *BMC Bioinformatics*, **8**, 132.

Allison, D. B., Cui, X., Page, G. P., and Sabripour, M. (2006). Microarray data analysis: from disarray to consolidation and consensus. *Nat Rev Genet*, **7**(1), 55–65.

Andreasson, E., Jenkins, T., Brodersen, P., Thorgrimsen, S., Petersen, N. H. T., Zhu, S. J., Qiu, J. L., Micheelsen, P., Rocher, A., Petersen, M., Newman, M. A., Nielsen, H. B., Hirt, H., Somssich, I., Mattsson, O., and Mundy, J. (2005). The MAP kinase substrate MKS1 is a regulator of plant defense responses. *EMBO J*, **24**(14), 2579–2589.

Archer, K. J., Dumur, C. I., Joel, S. E., and Ramakrishnan, V. (2006). Assessing quality of hybridized RNA in Affymetrix GeneChip experiments using mixed-effects models. *Biostatistics*, **7**(2), 198–212.

Ayroles, J. F. and Gibson, G. (2006). Analysis of variance of microarray data. *Method Enzymol*, **411**, 214–33.

Balasubramanian, S., Sureshkumar, S., Lempe, J., and Weigel, D. (2006). Potent induction of *Arabidopsis thaliana* flowering by elevated growth temperature. *PLoS Genet*, **2**(7), 980–989.

Baldi, P. and Long, A. D. (2001). A Bayesian framework for the analysis of microarray expression data: regularized t -test and statistical inferences of gene changes. *Bioinformatics*, **17**(6), 509–19.

Barrera, L., Benner, C., Tao, Y. C., Winzeler, E., and Zhou, Y. (2004). Leveraging two-way probe-level block design for identifying differential gene expression with high-density oligonucleotide arrays. *BMC Bioinformatics*, **5**, 42.

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *J Roy Stat Soc B*, **57**(1), 289–300.

Benjamini, Y. and Yekutieli, D. (2001). The control of the false discovery rate in multiple testing under dependency. *Ann Stat*, **29**(4), 1165–88.

Billingsley, P. (1986). *Probability and Measure*. Wiley series in probability and mathematical statistics. Wiley, New York.

Bolstad, B. M., Irizarry, R. A., Astrand, M., and Speed, T. P. (2003). A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, **19**(2), 185–93.

Bolstad, B. M., Collin, F., Simpson, K. M., Irizarry, R. A., and Speed, T. P. (2004). Experimental design and low-level analysis of microarray data. *Int Rev Neurobiol*, **60**, 25–58.

Bolstad, B. M., Irizarry, R. A., Gautier, L., and Wu, Z. (2005). Preprocessing high-density oligonucleotide arrays. In R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, and W. Huber, editors, *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, pages 13–32. Springer, New York.

Brodersen, P., Petersen, M., Nielsen, H. B., Zhu, S. J., Newman, M. A., Shokat, K. M., Rietz, S., Parker, J., and Mundy, J. (2006). Arabidopsis MAP kinase 4 regulates salicylic acid- and jasmonic acid/ethylene-dependent responses via EDS1 and PAD4. *Plant J*, **47**(4), 532–546.

Brown, H. and Prescott, R. (2006). *Applied Mixed Models in Medicine*. Wiley, Chichester, 2nd edition.

Chiu, T. Y. M., Leonard, T., and Tsui, K.-W. (1996). The matrix-logarithmic covariance model. *J Amer Statist Assoc*, **91**(433), 198–210.

Choe, S. E., Boutros, M., Michelson, A. M., Church, G. M., and Halfon, M. S. (2005). Preferred analysis methods for Affymetrix GeneChips revealed by a wholly defined control dataset. *Genome Biol*, **6**(2), R16.

Chu, T. M., Weir, B., and Wolfinger, R. (2002). A systematic statistical linear modeling approach to oligonucleotide array experiments. *Math Biosci*, **176**(1), 35–51.

Chu, T. M., Weir, B. S., and Wolfinger, R. D. (2004). Comparison of Li-Wong and loglinear mixed models for the statistical analysis of oligonucleotide arrays. *Bioinformatics*, **20**(4), 500–6.

Chudin, E., Walker, R., Kosaka, A., Wu, S. X., Rabert, D., Chang, T. K., and Kreder, D. E. (2002). Assessment of the relationship between signal intensities and transcript concentration for Affymetrix GeneChip arrays. *Genome Biol*, **3**(1), RESEARCH0005.

Churchill, G. A. (2002). Fundamentals of experimental design for cDNA microarrays. *Nat Genet*, **32 Suppl**, 490–5.

Cochran, W. G. (1934). The distribution of quadratic forms in a normal system, with applications to the analysis of variance. *Proc Camb Phil Soc*, **30**, 178–191.

Craig, A. T. (1943). Note on the independence of certain quadratic forms. *Ann Math Stat*, **14**, 195–197.

Cui, X. and Churchill, G. A. (2003). Statistical tests for differential expression in cDNA microarray experiments. *Genome Biol*, **4**(4), 210.

Cui, X., Hwang, J. T., Qiu, J., Blades, N. J., and Churchill, G. A. (2005). Improved statistical tests for differential gene expression by shrinking variance components estimates. *Biostatistics*, **6**(1), 59–75.

Dalma-Weiszhausz, D. D., Warrington, J., Tanimoto, E. Y., and Miyada, C. G. (2006). The Affymetrix GeneChip platform: an overview. *Method Enzymol*, **410**, 3–28.

David, H., Hofmann, G., Oliveira, A. P., Jarmer, H., and Nielsen, J. (2006). Metabolic network driven analysis of genome-wide transcription data from *Aspergillus nidulans*. *Genome Biol*, **7**(11).

Dawid, A. P. (1981). Some matrix-variate distribution theory: notational considerations and a Bayesian application. *Biometrika*, **68**(1), 265–274.

Dempster, A. P. (1958). A high dimensional two sample significance test. *Ann Math Stat*, **29**(4), 995–1010.

Dempster, A. P. (1960). A significance test for the separation of two highly multivariate small samples. *Biometrics*, **16**, 41–50.

Díaz-García, J. A. and González-Farías, G. (2005). Singular random matrix decompositions: Jacobians. *J Multivariate Anal*, **93**(2), 296–312.

Díaz-García, J. A. and Gutiérrez Jáimez, R. (1997). Proof of the conjectures of H. Uhlig on the singular multivariate beta and the Jacobian of a certain matrix transformation. *Ann Stat*, **25**(5), 2018–2023.

Díaz-García, J. A., Gutierrez Jáimez, R., and Mardia, K. V. (1997). Wishart and pseudo-Wishart distributions and some applications to shape theory. *J Multivariate Anal*, **63**(1), 73–87.

Dobbin, K., Shih, J. H., and Simon, R. (2003). Statistical design of reverse dye microarrays. *Bioinformatics*, **19**(7), 803–10.

Dudoit, S., Yang, Y. H., Callow, M. J., and Speed, T. P. (2002). Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. *Stat Sinica*, **12**, 111–39.

Duggan, D. J., Bittner, M., Chen, Y., Meltzer, P., and Trent, J. M. (1999). Expression profiling using cDNA microarrays. *Nat Genet*, **21**(1 Suppl), 10–4.

Dykstra, R. L. (1970). Establishing the positive definiteness of the sample covariance matrix. *Ann Math Stat*, **41**(6), 2153–2154.

Eaton, M. L. and Perlman, M. D. (1973). The non-singularity of generalized sample covariance matrices. *Ann Stat*, **1**, 710–717.

Eaves, I. A., Wicker, L. S., Ghandour, G., Lyons, P. A., Peterson, L. B., Todd, J. A., and Glynne, R. J. (2002). Combining mouse congenic strains and microarray gene expression analyses to study a complex trait: the NOD model of type 1 diabetes. *Genome Res*, **12**(2), 232–43.

Efron, B., Tibshirani, R., Storey, J. D., and Tusher, V. (2001). Empirical Bayes analysis of a microarray experiment. *J Am Stat Assoc*, **96**(456), 1151–1160.

Fan, J. B., Gunderson, K. L., Bibikova, M., Yeakley, J. M., Chen, J., Wickham Garcia, E., Lebruska, L. L., Laurent, M., Shen, R., and Barker, D. (2006). Illumina universal bead arrays. *Method Enzymol*, **410**, 57–73.

Firestein, G. S. and Pisetsky, D. S. (2002). DNA microarrays: boundless technology or bound by technology? Guidelines for studies using microarray technology. *Arthritis Rheum*, **46**(4), 859–61.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *J Am Stat Assoc*, **32**(200), 675–701.

Gautier, L., Moller, M., Friis-Hansen, L., and Knudsen, S. (2004). Alternative mapping of probes to genes for Affymetrix chips. *BMC Bioinformatics*, **5**, 111.

Gebicke-Haerter, P. (2005). Expression profiling methods used in drug abuse research. *Addict Biol*, **10**(1), 37–46.

Gentleman, R. C., Carey, V. J., Bates, D. M., Bolstad, B., Dettling, M., Dudoit, S., Ellis, B., Gautier, L., Ge, Y., Gentry, J., Hornik, K., Hothorn, T., Huber, W., Iacus, S., Irizarry, R., Leisch, F., Li, C., Maechler, M., Rossini, A. J., Sawitzki, G., Smith, C., Smyth, G., Tierney, L., Yang, J. Y., and Zhang, J. (2004). Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol*, **5**(10), R80.

Ghosh, M. and Sinha, B. K. (1980). On the robustness of least squares procedures in regression models. *J Multivariate Anal*, **10**(3), 332–342.

Giles, P. J. and Kipling, D. (2003). Normality of oligonucleotide microarray data and implications for parametric statistical analyses. *Bioinformatics*, **19**(17), 2254–62.

Graham, A. (1981). *Kronecker products and matrix calculus: with applications*. Ellis Horwood Series in Mathematics and its Applications. Ellis Horwood Ltd., Chichester.

Graybill, F. A. (1983). *Matrices with applications in statistics*. Wadsworth Statistics/Probability Series. Wadsworth Advanced Books and Software, Belmont, Calif., 2nd edition.

Gupta, A. K. and Nagar, D. K. (2000). *Matrix variate distributions*, volume 104 of *Chapman & Hall/CRC Monographs and Surveys in Pure and Applied Mathematics*. Chapman & Hall/CRC, Boca Raton, FL.

Hager, J. (2006). Making and using spotted DNA microarrays in an academic core laboratory. *Method Enzymol*, **410**, 135–68.

Hardiman, G. (2004). Microarray platforms–comparisons and contrasts. *Pharmacogenomics*, **5**(5), 487–502.

Harville, D. A. (1997). *Matrix algebra from a statistician's perspective*. Springer-Verlag, New York.

Hedenfalk, I., Duggan, D., Chen, Y., Radmacher, M., Bittner, M., Simon, R., Meltzer, P., Gusterson, B., Esteller, M., Kallioniemi, O. P., Wilfond, B., Borg, A., Trent, J., Raffeld, M., Yakhini, Z., Ben-Dor, A., Dougherty, E., Kononen, J., Bubendorf, L., Fehrle, W., Pittaluga, S., Gruvberger, S., Loman, N., Johannsson, O., Olsson, H., and Sauter, G. (2001). Gene-expression profiles in hereditary breast cancer. *New Engl J Med*, **344**(8), 539–48.

Henderson, H. V. and Searle, S. R. (1979). Vec and vech operators for matrices, with some uses in Jacobians and multivariate statistics. *Can J Stat*, **7**(1), 65–81.

Hoaglin, D. C., Mosteller, F., and Tukey, J. W. (2000). *Understanding Robust and Exploratory Data Analysis*. Wiley, New York.

Hsu, P. L. (1939). On the distribution of roots of certain determinantal equations. *Ann Eugenics*, **9**, 250–258.

Hu, H. and Olkin, I. (1991). A numerical procedure for finding the positive definite matrix closest to a patterned matrix. *Statist Probab Lett*, **12**(6), 511–515.

Hu, J. and Wright, F. A. (2007). Assessing differential gene expression with small sample sizes in oligonucleotide arrays using a mean-variance model. *Biometrics*, **63**(1), 41–9.

Hubbell, E., Liu, W. M., and Mei, R. (2002). Robust estimators for expression analysis. *Bioinformatics*, **18**(12), 1585–92.

Hueber, S. D., Bezdan, D., Henz, S. R., Blank, M., Wu, H. J., and Lohmann, I. (2007). Comparative analysis of Hox downstream genes in *Drosophila*. *Development*, **134**(2), 381–392.

Hughes, T. R., Marton, M. J., Jones, A. R., Roberts, C. J., Stoughton, R., Armour, C. D., Bennett, H. A., Coffey, E., Dai, H., He, Y. D., Kidd, M. J., King, A. M., Meyer, M. R., Slade, D., Lum, P. Y., Stepaniants, S. B., Shoemaker, D. D., Gachotte, D., Chakraburtty, K., Simon, J., Bard, M., and Friend, S. H. (2000). Functional discovery via a compendium of expression profiles. *Cell*, **102**(1), 109–26.

Irizarry, R., Wu, Z., and Jaffee, H. (2006). Comparison of Affymetrix GeneChip expression measures. *Bioinformatics*, **22**(7), 789–94.

Irizarry, R. A., Hobbs, B., Collin, F., Beazer-Barclay, Y. D., Antonellis, K. J., Scherf, U., and Speed, T. P. (2003). Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*, **4**(2), 249–64.

Jafari, P. and Azuaje, F. (2006). An assessment of recently published gene expression data analyses: reporting experimental design and statistical factors. *BMC Med Inform Decis Mak*, **6**, 27.

Jain, A. N., Tokuyasu, T. A., Snijders, A. M., Segraves, R., Albertson, D. G., and Pinkel, D. (2002). Fully automatic quantification of microarray image data. *Genome Res*, **12**(2), 325–32.

Jain, N., Thatte, J., Braciale, T., Ley, K., O'Connell, M., and Lee, J. K. (2003). Local-pooled-error test for identifying differentially expressed genes with a small number of replicated microarrays. *Bioinformatics*, **19**(15), 1945–51.

Jain, N., Cho, H., O'Connell, M., and Lee, J. K. (2005). Rank-invariant resampling based estimation of false discovery rate for analysis of small sample microarray data. *BMC Bioinformatics*, **6**(187), 187.

Johnson, R. A. and Wichern, D. W. (2002). *Applied Multivariate Statistical Analysis*. Prentice Hall, Upper Saddle River, NJ, 5th edition.

Karsten, S. L., Kudo, L. C., and Geschwind, D. H. (2004). Microarray platforms: introduction and application to neurobiology. *Int Rev Neurobiol*, **60**, 1–23.

Kendziorski, C. M., Newton, M. A., Lan, H., and Gould, M. N. (2003). On parametric empirical Bayes methods for comparing multiple groups using replicated gene expression profiles. *Stat Med*, **22**(24), 3899–914.

Kennedy, R. E., Archer, K. J., and Miles, M. F. (2006a). Empirical validation of the S-Score algorithm in the analysis of gene expression data. *BMC Bioinformatics*, **7**, 154.

Kennedy, R. E., Kerns, R. T., Kong, X., Archer, K. J., and Miles, M. F. (2006b). SScore: an R package for detecting differential gene expression without gene expression summaries. *Bioinformatics*, **22**(10), 1272–4.

Kerns, R. T., Zhang, L., and Miles, M. F. (2003). Application of the S-score algorithm for analysis of oligonucleotide microarrays. *Methods*, **31**, 274–81.

Kerr, M. K. and Churchill, G. A. (2001). Statistical design and the analysis of gene expression microarray data. *Genet Res*, **77**(2), 123–8.

Kerr, M. K., Martin, M., and Churchill, G. A. (2000). Analysis of variance for gene expression microarray data. *J Comput Biol*, **7**(6), 819–37.

Khatri, C. G. (1959). On the mutual independence of certain statistics. *Ann Math Stat*, **30**, 1258–1262.

Khatri, C. G. (1970). A note on Mitra's paper "A density-free approach to the matrix variate beta distribution". *Sankhyā Ser A*, **A32**, 311–218.

Koning, R. H., Neudecker, H., and Wansbeek, T. (1991). Block Kronecker products and the vecb operator. *Linear Algebra Appl*, **149**, 165–184.

Konno, Y. (1991). A note on estimating eigenvalues of scale matrix of the multivariate F-distribution. *Ann Inst Stat Math*, **43**, 157–165.

Le, N., Sun, L., and Zidek, J. V. (1998). A note on the existence of maximum likelihood estimates for Gaussian-inverted Wishart models. *Stat Prob Lett*, **40**(2), 133–137.

Lee, J. K. (2001). Analysis issues for gene expression array data. *Clin Chem*, **47**(8), 1350–1352.

Leibfried, A., To, J. P. C., Busch, W., Stehling, S., Kehle, A., Demar, M., Kieber, J. J., and Lohmann, J. U. (2005). WUSCHEL controls meristem function by direct regulation of cytokinin-inducible response regulators. *Nature*, **438**(7071), 1172–1175.

Lemieux, S. (2006). Probe-level linear model fitting and mixture modeling results in high accuracy detection of differential gene expression. *BMC Bioinformatics*, **7**, 391.

Lemon, W. J., Palatini, J. J., Krahe, R., and Wright, F. A. (2002). Theoretical and experimental comparisons of gene expression indexes for oligonucleotide arrays. *Bioinformatics*, **18**(11), 1470–6.

Lemon, W. J., Liyanarachchi, S., and You, M. (2003). A high performance test of differential gene expression for oligonucleotide arrays. *Genome Biol*, **4**(10), R67.

Li, C. and Hung Wong, W. (2001). Model-based analysis of oligonucleotide arrays: model validation, design issues and standard error application. *Genome Biol*, **2**(8), RESEARCH0032.

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

Lipshutz, R. J., Fodor, S. P., Gingeras, T. R., and Lockhart, D. J. (1999). High density synthetic oligonucleotide arrays. *Nat Genet*, **21**(1 Suppl), 20–4.

Liu, W. M., Mei, R., Di, X., Ryder, T. B., Hubbell, E., Dee, S., Webster, T. A., Harrington, C. A., Ho, M. H., Baid, J., and Smeekens, S. P. (2002). Analysis of high density expression microarrays with signed-rank call algorithms. *Bioinformatics*, **18**(12), 1593–9.

Liu, X., Milo, M., Lawrence, N. D., and Rattray, M. (2005). A tractable probabilistic model for Affymetrix probe-level analysis across multiple chips. *Bioinformatics*, **21**(18), 3637–44.

Liu, X. J., Milo, M., Lawrence, N. D., and Rattray, M. (2006). Probe-level measurement error improves accuracy in detecting differential gene expression. *Bioinformatics*, **22**(17), 2107–2113.

Liu, X. J., Lin, K. K., Andersen, B., and Rattray, M. (2007). Including probe-level uncertainty in model-based gene expression clustering. *BMC Bioinformatics*, **8**.

Lonnstedt, I. and Speed, T. (2002). Replicated microarray data. *Stat Sinica*, **12**, 31–46.

Mack, G. A. and Skillings, J. H. (1980). A Friedman-type rank test for main effects in a two-factor ANOVA. *J Am Stat Assoc*, **75**(372), 947–951.

Magnus, J. R. (1988). *Linear structures*, volume 42. Griffin, London.

Magnus, J. R. and Neudecker, H. (1979). The commutation matrix: some properties and applications. *Ann Stat*, **7**(2), 381–394.

Magnus, J. R. and Neudecker, H. (1980). The elimination matrix: some lemmas and applications. *SIAM J Algebra Discrete Meth*, **1**(4), 422–449.

Magnus, J. R. and Neudecker, H. (1985). Matrix differential calculus with applications to simple, Hadamard, and Kronecker products. *J Math Psychol*, **29**(4), 474–492.

Magnus, J. R. and Neudecker, H. (1999). *Matrix differential calculus with applications in statistics and econometrics*. Wiley Series in Probability and Statistics. John Wiley & Sons Ltd., Chichester.

Mansourian, R., Mutch, D. M., Antille, N., Aubert, J., Fogel, P., Goff, J.-M. L., Moulin, J., Petrov, A., Rytz, A., Voegel, J. J., and Roberts, M.-A. (2004). The Global Error Assessment (GEA) model for the selection of differentially expressed genes in microarray data. *Bioinformatics*, **20**(16), 2726–37.

Master, S. R., Stoddard, A. J., Bailey, L. C., Pan, T. C., Dugan, K. D., and Chodosh, L. A. (2005). Genomic analysis of early murine mammary gland development using novel probe-level algorithms. *Genome Biol*, **6**(2).

Mitra, S. K. (1970). A density-free approach to the matrix variate beta distribution. *Sankhyā Ser A*, **32**, 81–88.

Mogensen, J., Nielsen, H. B., Hofmann, G., and Nielsen, J. (2006). Transcription analysis using high-density micro-arrays of *Aspergillus nidulans* wild-type and *creA* mutant during growth on glucose or ethanol. *Fungal Genet Biol*, **43**(8), 593–603.

Muirhead, R. J. (1982). *Aspects of Multivariate Statistical Theory*. Wiley, New York.

Muirhead, R. J. and Verathaworn, T. (1985). On estimating the latent roots of $\Sigma_1\Sigma_2^{-1}$. In P. R. Krishnaiah, editor, *Multivariate Analysis*, volume VI, pages 431–447. North Holland, Amsterdam.

Myers, R. H. (1990). *Classical and modern regression with applications*. PWS-KENT, Boston, 2nd edition.

Nelder, J. A. and Mead, R. (1965). A simplex algorithm for function minimization. *Comput J*, **7**, 308–313.

Neudecker, H. (1968). The Kronecker matrix product and some of its applications in econometrics. *Stat Neerl*, **22**, 69–82.

Neudecker, H. (1969). Some theorems on matrix differentiation with special reference to Kronecker matrix products. *J Am Stat Assoc*, **64**(327), 953–963.

Neuhäuser, M. and Jöckel, K.-H. (2006). A bootstrap test for the analysis of microarray experiments with a very small number of replications. *Appl Bioinformatics*, **5**(3), 173–9.

Newton, M. A., Kendziorski, C. M., Richmond, C. S., Blattner, F. R., and Tsui, K. W. (2001). On differential variability of expression ratios: improving statistical inference about gene expression changes from microarray data. *J Comput Biol*, **8**(1), 37–52.

Nguyen, D. V., Arpat, A. B., Wang, N., and Carroll, R. J. (2002). DNA microarray experiments: biological and technological aspects. *Biometrics*, **58**(4), 701–17.

Nielsen, M. E., Lok, F., and Nielsen, H. B. (2006). Distinct developmental defense activations in barley embryos identified by transcriptome profiling. *Plant Mol Biol*, **61**(4-5), 589–601.

Olkin, I. and Rubin, H. (1964). Multivariate beta distributions and independence properties of the Wishart distribution. *Ann Math Stat*, **35**(1), 261–269.

Olkin, I. and Sampson, A. R. (1972). Jacobians of matrix transformations and induced functional equations. *Linear Algebra Appl*, **5**, 257–276.

Park, T., Kim, Y., Bekiranov, S., and Lee, J. K. (2007). Error-pooling-based statistical methods for identifying novel temporal replication profiles of human chromosomes observed by DNA tiling arrays. *Nucleic Acids Res*, **35**(9), e69.

Patterson, T. A., Lobenhofer, E. K., Fulmer-Smentek, S. B., Collins, P. J., Chu, T. M., Bao, W., Fang, H., Kawasaki, E. S., Hager, J., Tikhonova, I. R., Walker, S. J., Zhang, L., Hurban, P., de Longueville, F., Fuscoe, J. C., Tong, W., Shi, L., and Wolfinger, R. D. (2006). Performance comparison of one-color and two-color platforms within the MicroArray Quality Control (MAQC) project. *Nature Biotechnol*, **24**(9), 1140–50.

Pinheiro, J. C. and Bates, D. M. (2000). *Mixed-Effects Models in S and S-PLUS*. Springer, New York.

Purutcuoglu, V. and Wit, E. (2007). FGX: a frequentist gene expression index for Affymetrix arrays. *Biostatistics*, **8**(2), 433–437.

R Development Core Team (2007). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rattray, M., Liu, X. J., Sanguinetti, G., Milo, M., and Lawrence, N. D. (2006). Propagating uncertainty in microarray data analysis. *Brief Bioinform*, **7**(1), 37–47.

Reimers, M. (2005). Statistical analysis of microarray data. *Addict Biol*, **10**(1), 23–35.

Reiner, A., Yekutieli, D., and Benjamini, Y. (2003). Identifying differentially expressed genes using false discovery rate controlling procedures. *Bioinformatics*, **19**(3), 368–75.

Rencher, A. C. (2000). *Linear models in statistics*. Wiley Series in Probability and Statistics. John Wiley & Sons Inc., New York.

Rencher, A. C. (2002). *Methods of multivariate analysis*. Wiley Series in Probability and Statistics. Wiley-Interscience [John Wiley & Sons], New York, 2nd edition.

Resnick, S. I. (1999). *A Probability Path*. Birkhäuser, Boston.

Rosenwald, A., Wright, G., Chan, W. C., Connors, J. M., Campo, E., Fisher, R. I., Gascoyne, R. D., Muller-Hermelink, H. K., Smeland, E. B., Giltnane, J. M., Hurt, E. M., Zhao, H., Averett, L., Yang, L., Wilson, W. H., Jaffe, E. S., Simon, R., Klausner, R. D., Powell, J., Duffey, P. L., Longo, D. L., Greiner, T. C., Weisenburger, D. D., Sanger, W. G., Dave, B. J., Lynch, J. C., Vose, J., Armitage, J. O., Montserrat, E., Lopez-Guillermo, A., Grogan, T. M., Miller, T. P., LeBlanc, M., Ott, G., Kvaloy, S., Delabie, J., Holte, H., Krajci, P., Stokke, T., and Staudt, L. M. (2002). The use of molecular profiling to predict survival after chemotherapy for diffuse large-B-cell lymphoma. *New Engl J Med*, **346**(25), 1937–47.

Sanguinetti, G., Rattray, M., and Lawrence, N. D. (2006). A probabilistic dynamical model for quantitative inference of the regulatory mechanism of transcription. *Bioinformatics*, **22**(14), 1753–1759.

Sartor, M. A., Tomlinson, C. R., Wesselkamper, S. C., Sivaganesan, S., Leikauf, G. D., and Medvedovic, M. (2006). Intensity-based hierarchical Bayes method improves testing for differentially expressed genes in microarray experiments. *BMC Bioinformatics*, **7**, 538.

Schadt, E. E., Li, C., Ellis, B., and Wong, W. H. (2001). Feature extraction and normalization algorithms for high-density oligonucleotide gene expression array data. *J Cell Biochem Suppl*, **37**, 120–5.

Schena, M., Shalon, D., Davis, R., and Brown, P. (1995). Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, **270**, 467–70.

Schwab, R., Palatnik, J. F., Riester, M., Schommer, C., Schmid, M., and Weigel, D. (2005). Specific effects of microRNAs on the plant transcriptome. *Dev Cell*, **8**(4), 517–527.

Schwab, R., Ossowski, S., Riester, M., Warthmann, N., and Weigel, D. (2006). Highly specific gene silencing by artificial microRNAs in *Arabidopsis*. *Plant Cell*, **18**(5), 1121–1133.

Schwallie, D. P. (1985). Positive definite maximum likelihood covariance estimators. *Econom Lett*, **17**(1-2), 115–117.

Searle, S. R. (1978). A univariate formulation of the multivariate linear model. In *Contributions to survey sampling and applied statistics*, pages 181–189. Academic Press, New York.

Searle, S. R. (1982). *Matrix algebra useful for statistics*. Wiley Series in Probability and Mathematical Statistics: Applied Probability and Statistics. John Wiley & Sons Ltd., Chichester.

Shi, L., Shi, L., Reid, L. H., Jones, W. D., Shippy, R., Warrington, J. A., Baker, S. C., Collins, P. J., de Longueville, F., Kawasaki, E. S., Lee, K. Y., Luo, Y., Sun, Y. A., Willey, J. C., Setterquist, R. A., Fischer, G. M., Tong, W., Dragan, Y. P., Dix, D. J., Frueh, F. W., Goodsaid, F. M., Herman, D., Jensen, R. V., Johnson, C. D., Lobenhofer, E. K., Puri, R. K., Scherf, U., Thierry-Mieg, J., Wang, C., Wilson, M., Wolber, P. K., Zhang, L., Amur, S., Bao, W., Barbacioru, C. C., Lucas, A. B., Bertholet, V., Boysen, C., Bromley, B., Brown, D., Brunner, A., Canales, R., Cao, X. M., Cebula, T. A., Chen, J. J., Cheng, J., Chu, T. M., Chudin, E., Corson, J., Corton, J. C., Croner, L. J., Davies, C., Davison, T. S., Delenstarr, G., Deng, X., Dorris, D., Eklund, A. C., Fan, X. H., Fang, H., Fulmer-Smentek, S., Fuscoe, J. C., Gallagher, K., Ge, W., Guo, L., Guo, X., Hager, J., Haje, P. K., Han, J., Han, T., Harbottle, H. C., Harris, S. C., Hatchwell, E.,

Hauser, C. A., Hester, S., Hong, H., Hurban, P., Jackson, S. A., Ji, H., Knight, C. R., Kuo, W. P., Leclerc, J. E., Levy, S., Li, Q. Z., Liu, C., Liu, Y., Lombardi, M. J., Ma, Y., Magnuson, S. R., Maqsodi, B., McDaniel, T., Mei, N., Myklebost, O., Ning, B., Novoradovskaya, N., Orr, M. S., Osborn, T. W., Papallo, A., Patterson, T. A., Perkins, R. G., Peters, E. H., *et al.* (2006). The MicroArray Quality Control (MAQC) project shows inter- and intraplatform reproducibility of gene expression measurements. *Nature Biotechnol*, **24**(9), 1151–1161.

Simon, R., Radmacher, M. D., and Dobbin, K. (2002). Design of studies using DNA microarrays. *Genet Epidemiol*, **23**(1), 21–36.

Singh-Gasson, S., Green, R. D., Yue, Y., Nelson, C., Blattner, F., Sussman, M. R., and Cerrina, F. (1999). Maskless fabrication of light-directed oligonucleotide microarrays using a digital micromirror array. *Nature Biotechnol*, **17**(10), 974–8.

Smyth, G. K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Stat Appl Genet Mol Biol*, **3**, Article3.

Smyth, G. K. (2005). Limma: linear models for microarray data. In R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, and W. Huber, editors, *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, pages 397–420. Springer, New York.

Srivastava, M. S. (2003). Singular Wishart and multivariate beta distributions. *Ann Stat*, **31**(5), 1537–1560.

Srivastava, M. S. (2007). Multivariate theory for analyzing high dimensional data. *J Japan Statist Soc*, **37**(1), 53–86.

Srivastava, M. S. and Khatri, C. G. (1979). *An Introduction to Multivariate Statistics*. North-Holland, New York.

Srivastava, M. S. and von Rosen, D. (2002). Regression models with unknown singular covariance matrix. *Linear Algebra Appl*, **354**, 255–273.

Stalteri, M. A. and Harrison, A. P. (2007). Interpretation of multiple probe sets mapping to the same gene in Affymetrix GeneChips. *BMC Bioinformatics*, **8**, 13.

Tan, W. Y. (1969). Note on the multivariate and the generalized multivariate beta distributions. *J Am Stat Assoc*, **64**, 230–241.

Thompson, K. L., Rosenzweig, B. A., Pine, P. S., Retief, J., Turpaz, Y., Afshari, C. A., Hamadeh, H. K., Damore, M. A., Boedigheimer, M., Blomme, E., Ciurlionis, R., Waring, J. F., Fuscoe, J. C., Paules, R., Tucker, C. J., Fare, T., Coffey, E. M., He, Y., Collins, P. J., Jarnagin, K., Fujimoto, S., Ganter, B., Kiser, G., Kaysser-Kranich, T., Sina, J., and Sistare, F. D. (2005). Use of a mixed tissue RNA design for performance assessments on multiple microarray formats. *Nucleic Acids Res*, **33**(22), e187.

Tumor Analysis Best Practices Working Group (2004). Expression profiling–best practices for data generation and interpretation in clinical trials. *Nat Rev Genet*, **5**(3), 229–37.

Tusher, V. G., Tibshirani, R., and Chu, G. (2001). Significance analysis of microarrays applied to the ionizing radiation response. *Proc Natl Acad Sci U S A*, **98**, 5116–21.

Uhlig, H. (1994). On singular Wishart and singular multivariate beta distributions. *Ann Stat*, **22**(1), 395–405.

Van Gelder, R. N., von Zastrow, M. E., Yool, A., Dement, W. C., Barchas, J. D., and Eberwine, J. H. (1990). Amplified RNA synthesized from limited quantities of heterogeneous cDNA. *Proc Natl Acad Sci U S A*, **87**(5), 1663–7.

Weng, L., Dai, H., Zhan, Y., He, Y., Stepaniants, S. B., and Bassett, D. E. (2006). Rosetta error model for gene expression analysis. *Bioinformatics*, **22**(9), 1111–21.

Wigge, P. A., Kim, M. C., Jaeger, K. E., Busch, W., Schmid, M., Lohmann, J. U., and Weigel, D. (2005). Integration of spatial and temporal information during floral induction in Arabidopsis. *Science*, **309**(5737), 1056–1059.

Wishart, J. (1928). The generalized product moment distribution in samples from a normal multivariate population. *Biometrika*, **A20**, 32–52.

Wolber, P. K., Collins, P. J., Lucas, A. B., De Witte, A., and Shannon, K. W. (2006). The Agilent in situ-synthesized microarray platform. *Method Enzymol*, **410**, 28–57.

Wolfinger, R. D., Gibson, G., Wolfinger, E. D., Bennett, L., Hamadeh, H., Bushel, P., Afshari, C., and Paules, R. S. (2001). Assessing gene significance from cDNA microarray expression data via mixed models. *J Comput Biol*, **8**(6), 625–37.

Wright, G. W. and Simon, R. M. (2003). A random variance model for detection of differential gene expression in small microarray experiments. *Bioinformatics*, **19**(18), 2448–55.

Wu, Z. and Irizarry, R. A. (2005). Stochastic models inspired by hybridization theory for short oligonucleotide arrays. *J Comput Biol*, **12**(6), 882–93.

Wu, Z., Irizarry, R. A., Gentleman, R., Martinez-Murillo, F., and Spencer, F. (2004). A model-based background adjustment for oligonucleotide expression arrays. *J Am Stat Assoc*, **99**(468), 909–917.

Yang, I. V., Chen, E., Hasseman, J. P., Liang, W., Frank, B. C., Wang, S., Sharov, V., Saeed, A. I., White, J., Li, J., Lee, N. H., Yeatman, T. J., and Quackenbush, J. (2002). Within the fold: assessing differential expression measures and reproducibility in microarray assays. *Genome Biol*, **3**(11), research0062.

Yang, Y., Hoh, J., Broger, C., Neeb, M., Edington, J., Lindpaintner, K., and Ott, J. (2003). Statistical methods for analyzing microarray feature data with replications. *J Comput Biol*, **10**(2), 157–69.

Yang, Y. H. and Speed, T. (2002). Design issues for cDNA microarray experiments. *Nat Rev Genet*, **3**(8), 579–88.

Yang, Y. H., Buckley, M. J., and Speed, T. P. (2001). Analysis of cDNA microarray images. *Brief Bioinform*, **2**(4), 341–9.

Yekutieli, D. (2002). *Theoretical Results Needed for Appyling the False Discovery Rate in Statistical Problems*. Ph.D. thesis, Tel-Aviv University.

Zakharkin, S. O., Kim, K., Mehta, T., Chen, L., Barnes, S., Scheirer, K. E., Parrish, R. S., Allison, D. B., and Page, G. P. (2005). Sources of variation in Affymetrix microarray experiments. *BMC Bioinformatics*, **6**, 214.

Zhang, L., Wang, L., Ravindranathan, A., and Miles, M. F. (2002). A new algorithm for analysis of oligonucleotide arrays: Application to expression profiling in mouse brain regions. *J Mol Biol*, **317**, 225–35.

Appendix A

Source Code Listings

A.1 Quality Control Assessment of the Choe *et al.* Spike-in Dataset

```
#############################################################
#
# Program Name: ChoeQuality.pl
# Author: Richard Kennedy
# Date: 12/14/2007
#
# Purpose:  This program performs quality assessments for
# the Choe et al. Golden Spike dataset.
#
# Description: This program performs quality assessments for
# the Choe et al. Golden Spike dataset in two ways.  First,
# the number of clone IDs and probesets assigned to each
# pool number are computed from the dataset provided on the
# authors' website, which are compared to the table in their
# BMC Bioinformatics article.  Second, the probesets are
# mapped to pool numbers, then pool numbers to fold change,
# to give the fold change data for each probeset.  These
# fold change data are compared to the fold change data
# for each probeset provided on the authors' website.
#
#############################################################

# Define several hash tables for analyzing the data.  The
# hash data structure makes it particularly easy to track
# the assignments using the key value, which corresponds to
# the clone ID or the pool number.

# Create a hash table for the number of clones assigned to
# each pool.  Note that, in addition to the numbered pools,
# there are two additional pools (described in the Choe et
# al. article):  empty, which are not assigned to any pool
# (these are probesets that were not spiked in); and mixed,
# which are weakly assigned to multiple pools.  These are
# identified as such in the authors' datafiles.  The key
# value for the hash table is the pool number, and the
# associated data value is the number of clones, which
# is initialized to 0.  The actual number will be calculated
# later.
%CloneCount = (empty => 0,
               mixed => 0,
```

```
                1 => 0,
                2 => 0,
                3 => 0,
                4 => 0,
                5 => 0,
                6 => 0,
                7 => 0,
                8 => 0,
                9 => 0,
                10 => 0,
                13 => 0,
                14 => 0,
                15 => 0,
                16 => 0,
                17 => 0,
                18 => 0,
                19 => 0);

# Create another hash table for the fold change assigned to
# each pool.  These data are obtained from columns 1 and 4
# of Table 1 in the Choe et al. article.  The empty clones
# are arbitrarily assigned a concentration of -1 and the
# mixed clones a concentration -2 by the original authors
# for identification.  The key value for the hash is the
# pool number, and the associated data value is the fold
# change of the spike (S) relative to control (C).
%PoolFold = (empty => -1,
                mixed => -2,
                unassigned => -3,
                1 => 1.2,
                2 => 2,
                3 => 1.5,
                4 => 2.5,
                5 => 1.2,
                6 => 3,
                7 => 3.5,
                8 => 1.5,
                9 => 4,
                10 => 1.7,
                13 => 1,
                14 => 1,
                15 => 1,
                16 => 1,
                17 => 1,
                18 => 1,
```

```
                19 => 1);

# Create a third hash table for the number of probesets
# assigned to each pool.  These values are computed from the
# data files at the authors' website.  The values of -1 and
# -2 again represent the empty and mixed clones.  The value
# of -3 is used to record any probesets which were not
# assigned to any pool.  The key value for the hash table is
# the pool number, and the associated data value is the
# number of probesets.
%ProbesetCount = (empty => 0,
                  mixed => 0,
                  unassigned => 0,
                  1 => 0,
                  2 => 0,
                  3 => 0,
                  4 => 0,
                  5 => 0,
                  6 => 0,
                  7 => 0,
                  8 => 0,
                  9 => 0,
                  10 => 0,
                  13 => 0,
                  14 => 0,
                  15 => 0,
                  16 => 0,
                  17 => 0,
                  18 => 0,
                  19 => 0);

# Create an empty hash structure for storing the pool number
# assigned to each clone.  This will be filled using the
# data file from the authors' website.  The key value for
# the hash table will be the clone ID, and the associated
# data value will be the pool number.
%ClonePool = ();

# Create an empty hash structure for storing the Flybase ID
# number assigned to each clone.  This is not used in the
# analysis but is output in the data file for completeness.
# The key value for the hash table will be the clone ID, and
# the associated data value will be the Flybase ID.
%CloneFlybase = ();
```

```
# Create an empty hash structure for storing the gene name
# assigned to each clone.  This is not used in the analysis
# but is output in the data file for completeness.  The key
# value for the hash table will be the clone ID, and the
# associated data value will be the gene name.
%CloneGene = ();

# Create an empty two-way hash structure (a hash of hashes)
# for identifying the probesets assigned to each pool.  The
# first key value of the hash table will be the pool number,
# and the second key value of the hash table will be the
# probeset ID.  The associated data value is simply an
# indicator variable set to 1 if the probeset is in the
# specified pool.
%ProbesetPool = ();

# Create a list containing the pool numbers and all possible
# key assignments (pool numbers plus empty and mixed
# categories).  These will be used as indices for printing
# the data in the appropriate order (since Perl sorts by the
# ASCII collating sequence rather than numeric).
@PoolIndex =
   ('1','2','3','4','5','6','7','8','9','10','13','14','15',
   '16','17','18','19');

@AllIndex =
   ('1','2','3','4','5','6','7','8','9','10','13','14','15',
   '16','17','18','19','empty','mixed');

# Open the data file (from the authors' website) containing
# the information about each clone.  This is stored in
# a .csv file with the rows being the clone and the columns
# being the pool number, clone ID, Flybase ID, and gene
# name in order.
open(CHOE,"<gb-2005-6-2-r16-s8.csv");

# Loop to read the entire data file
while (<CHOE>) {

# Parse each line into fields
    chomp $_;
    @fields = split(/,/,$_);
    $Pool = $fields[0];
    $CloneID = $fields[1];
    $FlybaseID = $fields[2];
```

```
    $Gene = $fields[3];

# Increment the clone ID count for the specified pool number
    $CloneCount{$Pool} = $CloneCount{$Pool} + 1;

# Store each data item in the appropriate hash, indexed
# by clone ID for access
    $ClonePool{$CloneID} = $Pool;
    $CloneFlybase{$CloneID} = $FlybaseID;
    $CloneGene{$CloneID} = $Gene;
}
close(CHOE);

# Note that the clone IDs for the empty and mixed categories
# have only the pool and clone ID columns populated.  The
# concentration, Flybase ID, and gene data are set to
# arbitrary values so they can be identified later.
$ClonePool{'empty'} = 'empty';
$CloneFlybase{'empty'} = "";
$CloneGene{'empty'} = "";

$ClonePool{'mixed'} = 'mixed';
$CloneFlybase{'mixed'} = "";
$CloneGene{'mixed'} = "";

# Print a sorted list of the clone IDs showing the pool
# assignments.  This should reproduce column 2 of Table 1
# in the Choe et al. article.
print "Clone assignments to pools\n";
print "--------------------------\n";
print "Pool Number\tNumber of Clones\n";

# Loop through each of the key values, which are the pool
# numbers, and locate the corresponding number of
# clone IDs assigned to it
$TotalCount = 0;
foreach $key (@PoolIndex) {
    print $key,"\t\t",$CloneCount{$key},"\n";
    $TotalCount = $TotalCount + $CloneCount{$key};
}
print "--------------------------\n";
print "Total\t\t$TotalCount\n\n";

# Open the data file (from the authors' website) containing
# information about the mapping of the clones to Affymetrix
```

```
# ID numbers.  This is a tab-delimited text file with the
# rows being the probeset ID and the columns being the clone
# ID (there can be more than one clone ID per probeset, in
# which case the probeset is listed multiple times), the
# number of probe pairs in the probeset that match the clone
# ID, the fold change for the clone ID, and the fold change
# for the probeset (which is the weighted average of the
# clone ID fold change for probesets matching multiple clone
# IDs).
# Note that the original file on the website contains
# a header which was manually removed prior to running this
# analysis.  Also, the original text file on the web does
# not have the .txt extension and, depending on the Perl
# implementation, the end-of-line character may need to be
# changed to the Unix convention (line feed) for this code
# to work.
open(PROBESETMAP,"<mapping-affy2clones.txt");

# Create an output file showing the fold changes obtained
# in two different manners.  This will be a .csv file
# with rows being the probeset ID and the columns being
# the fold change computed directly (as a weighted average
# of the clone ID fold change in the file), the fold change
# computed indirectly (as a weighted average of the fold
# change of the pool to which the clone ID is assigned, which
# is from the previous input file), and the difference between
# the two.
open(OUTFILE,">ChoeMapping.csv");
print OUTFILE "Probe ID,Direct Fold Change,Indirect Fold
   Change,Difference\n";

# Initialize an accumulator variable for storing the total
# counts across all pools
$TotalCount = 0;

# Initialize the value of the previous probe read from the
# file to a null value, so that the start can be identified.
# This value is used to track the previous probe so that
# probesets mapping to multiple clone IDs can be
# appropriately processed
$PreviousProbeset = "";

# Loop through the entire input file
while (<PROBESETMAP>) {
```

```
# Increment the total count of the number of probesets
# (which includes duplicates)
    $TotalCount = $TotalCount + 1;

# Split the input line into fields
    chomp $_;
    @fields = split(/\t/,$_);
    $ProbesetID = $fields[0];
    $CloneID = $fields[1];
    $NumProbes = $fields[2];
    $CloneFold = $fields[3];
    $ProbesetFold = $fields[4];

# If this clone ID does not exist in the previously
# created pool hash, then there is no pool assignment
# for the clone ID, so print an error message.
    if (!defined($ClonePool{$CloneID})) {
        print "Note: Probeset $ProbesetID is assigned to Clone
           ID $CloneID,\n";
        print "      but Clone ID $CloneID has no pool
           assignment\n";
    } else {

# Otherwise, the clone ID does have a pool assignment.
# Get the pool number and add the current probeset ID
# to the list of probesets for this pool.
        $Pool = $ClonePool{$CloneID};
        $ProbesetPool{$Pool}{$ProbesetID} = 1;

# Check to see if this probeset ID is the same as the
# probeset ID from the previous line.  If so, add the
# weighted fold change and the total number of probes
# to the running total.
        if ($ProbesetID eq $PreviousProbeset) {
            $DirectFold = $DirectFold + $NumProbes * $CloneFold;
            $IndirectFold = $IndirectFold + $NumProbes *
               $PoolFold{$ClonePool{$CloneID}};
            $TotalProbes = $TotalProbes + $NumProbes;

# If this probeset ID is not the same as the probeset ID
# from the previous line, then all of the data for the
# previous probeset has been read in.  Compute the fold
# change for the previous probeset and output it to the file
        } else {
```

```
# If the previous probeset is not a null string, then this
# is not the start of the file.  Compute the fold change for
# the previous probeset by dividing the total by the number
# of probes for both the direct and indirect methods, as
# well as the difference between them.
            if ($PreviousProbeset ne "") {
                $DirectFold = $DirectFold / $TotalProbes;
                $IndirectFold = $IndirectFold / $TotalProbes;
                $Difference = abs($DirectFold - $IndirectFold);

# If the difference between the direct and indirect
# computations is not zero (within round-off error), then
# the clone ID has not been assigned the same fold change as
# the pool to which it belongs, so print an error message.
                if ($Difference > 1e-3) {
                    print "Note: Probeset ID $PreviousProbeset
                       has different fold change\n";
                    print "for direct vs. indirect
                       calculations.\n";
                    print "Direct Fold Change =
                       $DirectFold\tIndirect Fold Change =
                       $IndirectFold\n";
                }

# Write the data to the output file for all probesets, so
# that it may be reviewed later if desired.
                print OUTFILE "$PreviousProbeset,$DirectFold,
                   $IndirectFold,$Difference\n";
            }

# Save the data for the current probeset as the previous
# probeset, in preparation for reading the next line of
# data.
            $DirectFold = $NumProbes * $CloneFold;
            $IndirectFold = $NumProbes *
               $PoolFold{$ClonePool{$CloneID}};
            $TotalProbes = $NumProbes;
            $PreviousProbeset = $ProbesetID;
        }
    }

# Increment the count of the number of probesets for this
# pool
    $ProbesetCount{$Pool} = $ProbesetCount{$Pool} + 1;
}
```

```
# Note that the last line of the mapping data file has not
# been processed.  This sends the last line to the output
# file.
print OUTFILE "$PreviousProbeset,",$DirectFold/$TotalProbes,",",
    $IndirectFold/$TotalProbes,","$Difference\n";
print "\n\n";

close(PROBESETMAP);
close(OUTFILE);

# Print the list of pool numbers and the number of
# Affymetrix probesets assigned to each pool.
print "Probeset assignments to pools\n";
print "-----------------------------\n";
print "Pool Number\tNumber of Probesets\n";

# Loop through the key values, which are the pool numbers,
# and locate the corresponding number of probesets assigned
# to it
foreach $key (@AllIndex) {
    print $key,"\t\t",$ProbesetCount{$key},"\n";
}

# Show the total number of probesets across all pools
print "-----------------------------\n";
print "Total\t\t$TotalCount\n\n\n";

# Note that it is possible for a probeset to be assigned to
# a pool more than once if multiple clones in the same pool
# map to the same probeset.  This prints a listing of the
# pool numbers and the number of probesets assigned to each
# pool, counting each probeset only once per pool.  This
# should reproduce Table 1 in the Choe et al. article.
print "Unique probeset assignments to pools\n";
print "------------------------------------\n";
print "Pool Number\tNumber of Probesets\n";

# Loop through the key values, which are the pool numbers,
# and locate the corresponding data value, which is a hash
# table.
$TotalCount = 0;
for $key (@AllIndex) {

# The second hash table has key values that are the probeset
```

```
# IDs for all probesets assigned to the current pool number.
# These key values are unique, so the number of keys is the
# number of unique probesets assigned to that pool.
    @Probesetkeys = keys(%{$ProbesetPool{$key}});
    $ProbesetCount = @Probesetkeys;
    print $key,"\t\t",$ProbesetCount,"\n";
    $TotalCount = $TotalCount + $ProbesetCount;
}
print "-----------------------------------\n";
print "Total\t\t$TotalCount\n\n";
```

A.2 Logit-T Analysis of Spike-In Datasets

```
#########################################################
#
# Program Name: logitTAnalysis.R
# Author: Richard Kennedy
# Date: 12/20/2007
#
# Purpose:  This program performs an automated analysis on
# several sets of data using the Logit-T program function as
# the primary analysis tool.
#
# Description: This program analyzes three separate
# datasets, the Affymetrix U95 and U133 Latin Square and the
# GeneLogic Dilution data.  For each dataset, the
# appropriate data files are read and the logit-T scores
# computed.  One data file is created showing the logit-T
# for all of the probesets on the chip, in increasing order
# (or decreasing order of significance); one data file gives
# the number of spike-in probes (from both the original
# Affymetrix list and the expanded list of McGee et al.)
# that are highly ranked; and one data file shows the actual
# rank based on logit-T scores versus the expected rank
# based on the concentration fold-change from the
# spike-in data.  Although similar, separate computation
# routines are used for the Affymetrix U133 Latin Square,
# Affymetrix U95 Latin Square, and GeneLogic Dilution
# datasets due to slight differences in the analyses and for
# better readability.
#
#########################################################

# Load the affy library.  This is a standard library
# available through Bioconductor, which implements the
# functions for reading CEL files
library(affy)

#########################################################
#
# This performs an analysis of the Affymetrix U133 spike-in
# data set
#
#########################################################
```

```
# These are the filenames, which are stored in order of the
# ASCII collating sequence, as in the directory listing
fnames <- c("12_13_02_U133A_Mer_Latin_Square_Expt10_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt10_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt10_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt11_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt11_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt11_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt12_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt12_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt12_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt13_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt13_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt13_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt14_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt14_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt14_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt1_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt1_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt1_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt2_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt2_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt2_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt3_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt3_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt3_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt4_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt4_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt4_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt5_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt5_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt5_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt6_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt6_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt6_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt7_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt7_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt7_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt8_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt8_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt8_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt9_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt9_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt9_R3.CEL")
```

```
# Put the filenames in numerical order
fnames <- fnames[c(16:42,1:15)]

# These are the categories (experiment numbers) to which
# each of the chips belongs
category <- rep(1:14,each=3)

# These are the categories (experiment numbers) for the
# comparisons.  By default, the baseline condition will be
# experiment 1.  All experiments in the list will be
# compared to the baseline in turn
cat.names <- unique(category[category!=1])

# Create the groupfile for the logit-T program indicating
# which data files to analyze.  As per the documentation for
# the logit-T program, the groupfile is a tab-delimited text
# file, with the first column being the CEL file name, the
# second column designating the group membership, and the
# third column being a synonym (abbreviation) for the CEL
# file.  An example would be
#      1521a99hpp_av06.CEL     A     a1a06
#      1521b99hpp_av06.CEL     B     b1a06
# The first CEL file is 1521a99hpp_av06.CEL, which is part
# of group A. Its synonym, which is the name used in
# printouts, is a1a06.  The second CEL file, which is part
# of group B, is 1521b99hpp_av06.CEL and has synonym b1a06.
# The groups A and B will be compared to each other to
# generate logit-T scores.
#
# For this analysis, there are 14 groups, which will be
# labeled A-N.  Each group contains 3 CEL files, so group A
# will have synonyms A1, A2, A3, group B synonyms B1, B2,
# B3, and so on.
for (i in (1:13)) {
    index <- category==1 | category==cat.names[i]
    small.fnames <- fnames[index]
    group <- c(rep("A",3),rep(LETTERS[i+1],3))
    synonym <- paste(rep(c("A",LETTERS[i+1]),each=3),rep(1:
       3,2),sep="")
    data <- data.frame(small.fnames,group,synonym)
    write.table(data,file=paste("LogitTFoldU133Run",i,
       ".txt",sep=""),sep="\t",row.names=FALSE,col.names=
       FALSE,quote=FALSE)
}
```

```
# These are the names of the spiked-in clones for the
# original 42 probes reported by Affymetrix.  These are in
# the order given in the Affymetrix descriptor file included
# with the datasets.  Note that there are 3 clones in each
# group of clones spiked in at the same concentration for a
# given experiment (see the Affymetrix descriptor file for
# additional information).
spike.names <- c("203508_at", "204563_at", "204513_s_at",
   "204205_at", "204959_at", "207655_s_at", "204836_at",
   "205291_at", "209795_at", "207777_s_at", "204912_at",
   "205569_at", "207160_at", "205692_s_at", "212827_at",
   "209606_at", "205267_at", "204417_at", "205398_s_at",
   "209734_at", "209354_at", "206060_s_at", "205790_at",
   "200665_s_at", "207641_at", "207540_s_at", "204430_s_at",
   "203471_s_at", "204951_at", "207968_s_at", "AFFX-r2-TagA_at",
    "AFFX-r2-TagB_at", "AFFX-r2-TagC_at", "AFFX-r2-TagD_at",
   "AFFX-r2-TagE_at", "AFFX-r2-TagF_at", "AFFX-r2-TagG_at",
   "AFFX-r2-TagH_at", "AFFX-DapX-3_at", "AFFX-LysX-3_at",
   "AFFX-PheX-3_at", "AFFX-ThrX-3_at")

# These are the concentration data for the clones in each
# experiment.  These are ordered across columns by clone
# group and across rows by experiment (or chip group).
spike.conc <- matrix(data=
   c(0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,
   0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,0,
   0.25,0.5,1,2,4,8,16,32,64,128,256,512,0,0.125,
   0.5,1,2,4,8,16,32,64,128,256,512,0,0.125,0.25,
   1,2,4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,
   2,4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,
   4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,
   8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,
   16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,
   32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,
   64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,
   128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,
   256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,
   512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256), nrow=14,
   byrow=TRUE)

# These are the group numbers for each of the spiked-in
# clones given in the spike.names variable
spike.group <- rep(1:14,each=3)
names(spike.group) <- spike.names
```

```
# These are the names of the spiked-in clones for the
# expanded set of 64 probes reported by McGee et al.  These
# are in the order given in their article and the
# supplemental files.  Note that there are no longer 3
# clones in each group when using the expanded set.
expanded.spike <- c("200665_s_at", "203471_s_at", "203508_at",
   "204205_at", "204417_at", "204430_s_at", "204513_s_at",
   "204563_at", "204836_at", "204912_at", "204951_at",
   "204959_at", "205267_at", "205291_at", "205398_s_at",
   "205569_at", "205692_s_at", "205790_at", "206060_s_at",
   "207160_at", "207540_s_at", "207641_at", "207655_s_at",
   "207777_s_at", "207968_s_at", "208010_s_at", "209354_at",
   "209374_s_at", "209606_at", "209734_at", "209795_at",
   "212827_at", "AFFX-DapX-3_at", "AFFX-DapX-5_at",
   "AFFX-DapX-M_at", "AFFX-LysX-3_at", "AFFX-LysX-5_at",
   "AFFX-LysX-M_at", "AFFX-PheX-3_at", "AFFX-PheX-5_at",
   "AFFX-PheX-M_at", "AFFX-ThrX-3_at", "AFFX-ThrX-5_at",
   "AFFX-ThrX-M_at", "AFFX-r2-Bs-dap-3_at",
   "AFFX-r2-Bs-dap-5_at", "AFFX-r2-Bs-dap-M_at",
   "AFFX-r2-Bs-lys-3_at", "AFFX-r2-Bs-lys-5_at",
   "AFFX-r2-Bs-lys-M_at", "AFFX-r2-Bs-phe-3_at",
   "AFFX-r2-Bs-phe-5_at", "AFFX-r2-Bs-phe-M_at",
   "AFFX-r2-Bs-thr-3_s_at", "AFFX-r2-Bs-thr-5_s_at",
   "AFFX-r2-Bs-thr-M_s_at", "AFFX-r2-TagA_at",
   "AFFX-r2-TagB_at", "AFFX-r2-TagC_at", "AFFX-r2-TagD_at",
   "AFFX-r2-TagE_at", "AFFX-r2-TagF_at", "AFFX-r2-TagG_at",
   "AFFX-r2-TagH_at")

# These are the group numbers for each of the spiked-in
# clones given in the expanded.spike variable.  Positive
# numbers denote the original 42 clones reported by
# Affymetrix, negative numbers the supplemental 22 clones
# given by McGee et al., to facilitate separate analyses if
# necessary.
expanded.group <- c(8,10,1,2,6,9,1,1,3,4,10,2,6,3,7,4,5,8,8,
   5,9,9,2,4,10,-8,7,-5,6,7,3,5,13,-13,-13,14,-14,-14,14,-14,
   -14,14,-14,-14,-13,-13,-13,-14,-14,-14,-14,-14,-14,-14,-14,
   -14,11,11,11,12,12,12,13,13)
names(expanded.group) <- expanded.spike

# These are the number of probes in the expanded and
# original list of spiked-in clones.
num.large <- 64
num.small <- 42
```

```
# loop through the list of experiments for comparison
for (i in 1:length(cat.names)) {

# Get the subset of the CEL files used in this analysis
    index <- category==1 | category==cat.names[i]
    small.fnames <- fnames[index]

# Since the logit-T program is a compiled C program, the
# command-line text must be generated and executed via the
# system command.  As per the documentation for the logit-T
# program, the syntax for the command-line invocation is
#      logitT groupFile.txt celfiles cdffile outputprefix
# where groupFile.txt is the groupfile that was previously
# generated, celfiles is a list of the CEL files to be
# analyzed, cdffile is the CDF file (Chip Definition File,
# which contains information about the layout of the chip),
# and outputprefix is the output prefix that will be
# prepended to all data files generated by the logit-T
# program.
    prefix <- paste("LogitTFold",i,"U133",sep="")
    fname <- paste("LogitTFoldRun",i,".txt",sep="")
    cmd <- paste("/Users/rkennedy/logitT",fname,
       paste(small.fnames,collapse=" "),"HG-U133A_tag.CDF",
       prefix)
    system(cmd)

# Get the results of the logit-T analysis.  As per the
# documentation for the logit-T program, the following
# output files are produced, with the output prefix
# prepended to all filenames instead of xxxx:
#      xxxxT.txt is the main output file with the t-test
#                values for each probeset.  Rows are
#                arranged by probeset and columns are
#                arranged by comparison.
#      xxxxCV.txt is the file containing the coefficients of
#                 variation for each probeset.  This file is
#                 not used in the present analysis.
#      xxxxW.txt is the file containing the standardized
#                Wilcoxon W statistics for each probeset.
#                This file is not used in the present
#                analysis.
#      xxxx.theta is the file containing the LogitExp gene
#                 expression index described in the Lemon et
#                 al. article.  This file is not used in the
#                 present analysis.
```

```
# All files are tab-delimited text files.  For this
# analysis, each xxxxT.txt file contains only one
# comparison, so the files are two columns with the first
# being the probeset name and the second being the logit-T
# value for that comparison, along with a header in the
# first line of the file.
    fname <- paste("LogitTFold",i,"U133T.txt",sep="")
    results <- read.table(file=fname,header=TRUE,sep="\t")
    results <- results[-1,]
    gn <- results[,1]
    score <- results[,2]

    abs.score <- abs(score)
    index <- order(abs.score,decreasing=TRUE)

# rank the scores in decreasing order, with ties being
# assigned rank equal to the smallest rank of the group of
# ties
    ranking <- rank(abs.score,ties.method="min")

# reverse the rankings, as small logit-T scores indicate
# higher probability of differential expression
    ranking <- max(ranking) - ranking + 1
    ranking <- rep(1:length(unique(ranking)),times=
       as.vector(table(ranking)))
    names(ranking) <- gn[index]

# create a data frame with the probeset (gene) names, rank,
# and score in order of increasing logit-T scores
    results <- data.frame(name=gn[index],iteration=rep(i,
       length(index)),rank=ranking,score=abs.score[index])
    outfile <- paste("LogitTFoldOverallU133.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

# count the number of spike-in probes ranked in the top 42
# (for the original list) or top 64 (for the expanded list)
# of probesets
    small.count <- sum(!is.na(match(names(ranking)[1:
       num.small],spike.names)))
    large.count <- sum(!is.na(match(names(ranking)[1:
       num.large],expanded.spike)))
    results <- data.frame(iteration=i,small.count, large.count)
    outfile <- paste("LogitTFoldCountU133.csv",sep="")
```

```
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

# create a data frame with the expected rank (based on fold
# change of the spike-in concentration) of the expanded list
# of spike-in probesets to compare to the actual rank (based
# on the logit-T values)
    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    outfile <- paste("LogitTFoldRankU133.csv",sep="")
    results <- data.frame(name=expanded.spike,iteration=
       rep(i,length(expanded.spike)),expectedrank=
       fold.rank[abs(expanded.group[expanded.spike])],
       actualrank=ranking[expanded.spike])
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))
}

# End of Affymetrix U133 Analysis

############################################################
#
# This performs an analysis of the Affymetrix U95 spike-in
# data set.
#
# As this analysis is similar to the U133 analysis, only
# differences between the analyses will be highlighted.
#
############################################################

fnames <- c("1521a99hpp_av06.CEL", "1521b99hpp_av06.CEL",
   "1521c99hpp_av06.CEL", "1521d99hpp_av06.CEL",
   "1521e99hpp_av06.CEL", "1521f99hpp_av06.CEL",
   "1521g99hpp_av06.CEL", "1521h99hpp_av06.CEL",
   "1521i99hpp_av06.CEL", "1521j99hpp_av06.CEL",
   "1521k99hpp_av06.CEL", "1521l99hpp_av06r.CEL",
   "1521m99hpp_av06.CEL", "1521n99hpp_av06.CEL",
   "1521o99hpp_av06.CEL", "1521p99hpp_av06.CEL",
   "1521q99hpp_av06.CEL", "1521r99hpp_av06.CEL",
   "1521s99hpp_av06.CEL", "1521t99hpp_av06.CEL",
   "1532a99hpp_av04.CEL", "1532b99hpp_av04.CEL",
   "1532c99hpp_av04.CEL", "1532d99hpp_av04.CEL",
   "1532e99hpp_av04.CEL", "1532f99hpp_av04.CEL",
```

```
   "1532g99hpp_av04.CEL", "1532h99hpp_av04.CEL",
   "1532i99hpp_av04.CEL", "1532j99hpp_av04.CEL",
   "1532k99hpp_av04.CEL", "1532l99hpp_av04.CEL",
   "1532m99hpp_av04.CEL", "1532n99hpp_av04.CEL",
   "1532o99hpp_av04.CEL", "1532p99hpp_av04.CEL",
   "1532q99hpp_av04.CEL", "1532r99hpp_av04.CEL",
   "1532s99hpp_av04.CEL", "1532t99hpp_av04r.CEL",
   "2353a99hpp_av08.CEL", "2353b99hpp_av08r.CEL",
   "2353d99hpp_av08.CEL", "2353e99hpp_av08.CEL",
   "2353f99hpp_av08.CEL", "2353g99hpp_av08.CEL",
   "2353h99hpp_av08.CEL", "2353i99hpp_av08.CEL",
   "2353j99hpp_av08.CEL", "2353k99hpp_av08.CEL",
   "2353l99hpp_av08.CEL", "2353m99hpp_av08.CEL",
   "2353n99hpp_av08.CEL", "2353o99hpp_av08.CEL",
   "2353p99hpp_av08.CEL", "2353q99hpp_av08.CEL",
   "2353r99hpp_av08.CEL", "2353s99hpp_av08.CEL",
   "2353t99hpp_av08.CEL")

spike.names <- c("37777_at", "684_at", "1597_at", "38734_at",
   "39058_at", "36311_at", "36889_at", "1024_at", "36202_at",
   "36085_at", "40322_at", "407_at", "1091_at", "1708_at")

spike.conc <- matrix(data=
   c(0,0.25,0.5,1,2,4,8,16,32,64,128,0,512,1024,
   0.25,0.5,1,2,4,8,16,32,64,128,256,0.25,1024,0,
   0.5,1,2,4,8,16,32,64,128,256,512,0.5,0,0.25,
   1,2,4,8,16,32,64,128,256,512,1024,1,0.25,0.5,
   2,4,8,16,32,64,128,256,512,1024,0,2,0.5,1,
   4,8,16,32,64,128,256,512,1024,0,0.25,4,1,2,
   8,16,32,64,128,256,512,1024,0,0.25,0.5,8,2,4,
   16,32,64,128,256,512,1024,0,0.25,0.5,1,16,4,8,
   32,64,128,256,512,1024,0,0.25,0.5,1,2,32,8,16,
   64,128,256,512,1024,0,0.25,0.5,1,2,4,64,16,32,
   128,256,512,1024,0,0.25,0.5,1,2,4,8,128,32,64,
   256,512,1024,0,0.25,0.5,1,2,4,8,16,256,64,128,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512), ncol=14,
   byrow=TRUE)
```

```
spike.group <- 1:14
names(spike.group) <- spike.names

# These are the categories (experiment numbers) for the
# comparisons.  Note that one chip in Experiment 3 of the
# U95 dataset did not hybridize properly, so that there are
# only 2 chips in this comparison rather than 3.  Though
# not originally intended, this allows the assessment of the
# algorithms when differing numbers of chips are compared.
category <- c(1:20,1:20,(1:20)[-3])
cat.names <- unique(category[category!=1])

num.large <- 14
num.small <- 14

for (i in (1:length(cat.names))) {
    rep1 <- sum(category==1)
    rep2 <- sum(category==cat.names[i])
    small.fnames <- c(fnames[category==1],fnames[category==
       cat.names[i]])
    group <- c(rep("A",rep1),rep(LETTERS[i+1],rep2))
    synonym <- paste(group,c(1:rep1,1:rep2),sep="")
    data <- data.frame(small.fnames,group,synonym)
    write.table(data,file=paste("LogitTFoldU95Run",i,
       ".txt",sep=""),sep="\t",row.names=FALSE,col.names=
       FALSE,quote=FALSE)
}

for (i in 1:length(cat.names)) {
    index <- category==1 | category==cat.names[i]
    small.fnames <- fnames[index]
    prefix <- paste("LogitTFold",i,"U95",sep="")
    fname <- paste("LogitTFoldU95Run",i,".txt",sep="")
    cmd <- paste("/Users/rkennedy/logitT",fname,
       paste(small.fnames,collapse=" "), "HG_U95A.CDF", prefix)
    system(cmd)
    fname <- paste("LogitTFold",i,"U95T.txt",sep="")
    results <- read.table(file=fname,header=TRUE,sep="\t")
    results <- results[-1,]
    gn <- results[,1]
    score <- results[,2]
    abs.score <- abs(score)
    index <- order(abs.score,decreasing=TRUE)
    ranking <- rank(abs.score,ties.method="min")
    ranking <- max(ranking) - ranking + 1
```

```
    ranking <- rep(1:length(unique(ranking)),times=
       as.vector(table(ranking)))
    names(ranking) <- gn[index]
    results <- data.frame(name=gn[index],iteration=rep(i,
       length(index)),rank=ranking,score=abs.score[index])
    outfile <- paste("LogitTFoldOverallU95.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))
    small.count <- sum(!is.na(match(names(ranking)[1:
       num.small],spike.names)))
    results <- data.frame(iteration=i,small.count)
    outfile <- paste("LogitTFoldCountU95.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))
    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    results <- data.frame(name=spike.names,iteration=rep(i,
       length(spike.names)),expectedrank=
       fold.rank[spike.group[spike.names]],actualrank=
       ranking[spike.names])
    outfile <- paste("LogitTFoldRankU95.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

}

# End of Affymetrix U95 analysis

############################################################
#
# This performs an analysis on the GeneLogic Dilution data
# set.
#
# As this analysis is similar to the U133 and U95 analyses,
# only differences between the analyses will be highlighted.
#
############################################################

fnames <- c("92453hgu95a11.cel", "92454hgu95a11.cel",
   "92455hgu95a11.cel", "92456hgu95a11.cel",
   "92457hgu95a11.cel", "92458hgu95a11.cel",
   "92459hgu95a11.cel", "92460hgu95a11.cel",
   "92461hgu95a11.cel", "92462hgu95a11.cel",
```

```
   "92463hgu95a11.cel", "92464hgu95a11.cel",
   "92465hgu95a11.cel", "92466hgu95a11.cel",
   "92491hgu95a11.cel", "92492hgu95a11.cel",
   "92493hgu95a11.cel", "92494hgu95a11.cel",
   "92495hgu95a11.cel", "92496hgu95a11.cel",
   "92497hgu95a11.cel", "92498hgu95a11.cel",
   "92499hgu95a11.cel", "92500hgu95a11.cel",
   "92501hgu95a11.cel", "92503hgu95a11.cel")

fnames <- fnames[c(14,15,16,17,18,19,20,1,21,2,3,22,4,5,23,
   6,7,24,8,9,25,10,11,12,13,26)]

spike.conc <- matrix(data=c(0,0,0,0,0,0,0,0,0,0,
   0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,
   0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,
   1,1,1,1,1,1,1,1,1,1, 1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,
    2,2,2,2,2,2,2,2,2,2, 3,3,3,3,3,3,3,3,3,3,
   5,5,5,5,5,5,5,5,5,5,
   12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,
   25,25,25,25,25,25,25,25,25,25, 50,50,50,50,50,50,50,50,50,50,
    75,75,75,75,75,75,75,75,75,75,
   100,100,100,100,100,100,100,100,100,100,
   150,150,150,150,150,150,150,150,150,150), nrow=14,
   byrow=TRUE)

spike.names <- c("AFFX-BioB-5_at", "AFFX-BioB-M_at",
   "AFFX-BioB-3_at", "AFFX-BioC-5_at", "AFFX-BioC-3_at",
   "AFFX-BioDn-3_at", "AFFX-DapX-5_at", "AFFX-DapX-M_at",
   "AFFX-DapX-3_at", "AFFX-CreX-5_at")

spike.group <- 1:11
names(spike.group) <- spike.names

expanded.spike <- spike.names

expanded.group <- spike.group
names(expanded.group) <- expanded.spike

small.fnames <- fnames

category <- c(1,2,3,4,5,6,7,8,8,rep(9:12,each=3),13,13,14,14,14)

# These are the categories (experiment numbers) for the
# comparisons.  Note that only experiments 9 through 12 and
# experiment 14 have a sufficient number of chips for
```

```
# comparisons using all algorithms.  Thus, the baseline
# condition will be experiment 9.  All experiments in the
# list will be compared to the baseline in turn.
cat.names <- unique(category[category > 9])

num.large <- 10
num.small <- 10

for (i in 1:length(cat.names)) {
    rep1 <- sum(category==9)
    rep2 <- sum(category==cat.names[i])
    small.fnames <- c(fnames[category==9],fnames[category==
       cat.names[i]])
    group <- c(rep(LETTERS[1],rep1),rep(LETTERS[i+1],rep2))
    synonym <- paste(group,c(1:rep1,1:rep2),sep="")
    data <- data.frame(small.fnames,group,synonym)
    write.table(data,file=paste("LogitTFoldGDilutionRun",i,
       ".txt",sep=""),sep="\t",row.names=FALSE,col.names=
       FALSE,quote=FALSE)
}

for (i in 1:length(cat.names)) {
    index <- category==9 | category==cat.names[i]
    small.fnames <- fnames[index]
    prefix <- paste("LogitTFold",i,"GDilution",sep="")
    fname <- paste("LogitTFoldGDilutionRun",i,".txt",sep="")
    cmd <- paste("/Users/rkennedy/logitT",fname,
       paste(small.fnames,collapse=" "), "HG_U95A.CDF", prefix)
    code <- system(cmd)
    fname <- paste("LogitTFold",i,"GDilutionT.txt",sep="")
    results <- read.table(file=fname,header=TRUE,sep="\t")
    results <- results[-1,]
    gn <- results[,1]
    score <- results[,2]
    abs.score <- abs(score)
    index <- order(abs.score,decreasing=TRUE)

    ranking <- rank(abs.score,ties.method="min")
    ranking <- max(ranking) - ranking + 1
    ranking <- rep(1:length(unique(ranking)),times=
       as.vector(table(ranking)))
    names(ranking) <- gn[index]
    results <- data.frame(name=gn[index],iteration=rep(i,
       length(index)),rank=ranking,score=abs.score[index])
    outfile <- paste("LogitTFoldOverallGDilution.csv", sep="")
```

```
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))
    t.score <- 1-pt(abs.score,df=length(index)-2)
    posindex <- (t.score <= 0.001)

    small.count <- sum(!is.na(match(gn[posindex], spike.names)))
    large.count <- sum(!is.na(match(gn[posindex],
       expanded.spike)))
    truepos <- large.count
    falsepos <- sum(posindex) - large.count
    falseneg <- num.large - large.count
    trueneg <- (length(abs.score) - sum(posindex)) - falseneg
    results <- data.frame(iteration=i,truepos,falsepos,
       trueneg,falseneg)
    outfile <- paste("LogitTFoldCountGDilution.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    outfile <- paste("LogitTFoldRankGDilution.csv",sep="")
    results <- data.frame(name=expanded.spike,iteration=rep(i,
       length(expanded.spike)),expectedrank=
       fold.rank[abs(expanded.group[expanded.spike])],
       actualrank=ranking[expanded.spike])
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

}

# End of GeneLogic Dilution Analysis
```

A.3 mmgMOS Analysis of Spike-In Datasets

```
#############################################################
#
# Program Name: mmgmosAnalysis.R
# Author: Richard Kennedy
# Date: 12/21/2007
#
# Purpose:  This program performs an automated analysis on
# several sets of data using the RMA algorithm as the
# expression summary measure.
#
# Description: This program analyzes three separate
# datasets, the Affymetrix U95 and U133 Latin Square and the
# GeneLogic Dilution data.  For each dataset, the
# appropriate data files are read and the mmgmos expression
# summary computed.  The mmgmos values are then compared
# using multiple t-tests to give measures of significance
# for differential gene expression.  One data file is
# created showing the p-values of the t-tests for all of the
# probesets on the chip, in increasing order (or decreasing
# order of significance); one data file gives the number of
# spike-in probes (from both the original Affymetrix list
# and the expanded list of McGee et al.) that are highly
# ranked; and one data file shows the actual rank based on
# the p-values versus the expected rank based on the
# concentration fold-change from the spike-in data.
# Although similar, separate computation routines are used
# for the Affymetrix U133 Latin Square, Affymetrix U95 Latin
# Square, and GeneLogic Dilution datasets due to slight
# differences in the analyses and for better readability.
#
#############################################################

# Load the affy library.  This is a standard library
# available through Bioconductor, which implements the
# functions for reading CEL files
library(affy)

# Load the puma library.  This is a standard library
# available through Bioconductor, which implements the
# mmgmos function
library(puma)
```

```
# Load the multtest library.  This is a standard library
# available through Bioconductor, which implements the
# multiple t-test among other functions.
library(multtest)

# This function implements a pooled degrees of freedom
# function, which computes a composite degrees of freedom
# for a two-sample comparison based on the relative size
# of each of the two samples.
# Input: exprs - a matrix containing the expression values
#        compare - a vector denoting the condition of each
#                  column in the exprs matrix, with 0 denoting
#                  the baseline condition and 1 denoting the
#                  experimental condition
# Output: a vector containing the pooled degrees of freedom
#         for each row of the exprs matrix
df <- function(exprs,compare) {
    var1 <- var(exprs[compare==1])
    var0 <- var(exprs[compare==0])
    n1 <- length(exprs[compare==1])
    n0 <- length(exprs[compare==0])
    result <- (var1+var0)^2 / (var1^2/(n1-1)+var0^2/(n0-1))
    return(result)
}

# End of declared functions

############################################################
#
# This performs an analysis of the Affymetrix U133 spike-in
# data set
#
############################################################

# These are the filenames, which are stored in order of the
# ASCII collating sequence, as in the directory listing.
fnames <- c("12_13_02_U133A_Mer_Latin_Square_Expt10_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt10_R2.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt10_R3.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt11_R1.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt11_R2.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt11_R3.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt12_R1.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt12_R2.CEL",
```

```
 "12_13_02_U133A_Mer_Latin_Square_Expt12_R3.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt13_R1.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt13_R2.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt13_R3.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt14_R1.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt14_R2.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt14_R3.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt1_R1.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt1_R2.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt1_R3.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt2_R1.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt2_R2.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt2_R3.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt3_R1.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt3_R2.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt3_R3.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt4_R1.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt4_R2.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt4_R3.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt5_R1.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt5_R2.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt5_R3.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt6_R1.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt6_R2.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt6_R3.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt7_R1.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt7_R2.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt7_R3.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt8_R1.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt8_R2.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt8_R3.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt9_R1.CEL",
 "12_13_02_U133A_Mer_Latin_Square_Expt9_R2.CEL",
    "12_13_02_U133A_Mer_Latin_Square_Expt9_R3.CEL")

# Put the filenames in numerical order
fnames <- fnames[c(16:42,1:15)]

# These are the names of the spiked-in clones for the
# original 42 probes reported by Affymetrix.  These are in
# the order given in the Affymetrix descriptor file included
# with the dataseta.  Note that there are 3 clones in each
# group of clones spiked in at the same concentration for a
# given experiment (see the Affymetrix descriptor file for
# additional information).
```

```
spike.names <- c("203508_at", "204563_at", "204513_s_at",
   "204205_at", "204959_at", "207655_s_at", "204836_at",
   "205291_at", "209795_at", "207777_s_at", "204912_at",
   "205569_at", "207160_at", "205692_s_at", "212827_at",
   "209606_at", "205267_at", "204417_at", "205398_s_at",
   "209734_at", "209354_at", "206060_s_at", "205790_at",
   "200665_s_at", "207641_at", "207540_s_at", "204430_s_at",
   "203471_s_at", "204951_at", "207968_s_at", "AFFX-r2-TagA_at",
    "AFFX-r2-TagB_at", "AFFX-r2-TagC_at", "AFFX-r2-TagD_at",
   "AFFX-r2-TagE_at", "AFFX-r2-TagF_at", "AFFX-r2-TagG_at",
   "AFFX-r2-TagH_at", "AFFX-DapX-3_at", "AFFX-LysX-3_at",
   "AFFX-PheX-3_at", "AFFX-ThrX-3_at")

# These are the concentration data for the clones in each
# experiment.  These are ordered across columns by clone
# group and across rows by experiment (or chip group).
spike.conc <- matrix(data=
   c(0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,
   0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,0,
   0.25,0.5,1,2,4,8,16,32,64,128,256,512,0,0.125,
   0.5,1,2,4,8,16,32,64,128,256,512,0,0.125,0.25,
   1,2,4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,
   2,4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,
   4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,
   8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,
   16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,
   32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,
   64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,
   128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,
   256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,
   512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256), nrow=14,
   byrow=TRUE)

# These are the group numbers for each of the spiked-in
# clones given in the spike.names variable
spike.group <- rep(1:14,each=3)
names(spike.group) <- spike.names

# These are the names of the spiked-in clones for the
# expanded set of 64 probes reported by McGee et al.  These
# are in the order given in their article and the
# supplemental files.  Note that there are no longer 3
# clones in each group when using the expanded set.
expanded.spike <- c("200665_s_at", "203471_s_at", "203508_at",
   "204205_at", "204417_at", "204430_s_at", "204513_s_at",
```

```
   "204563_at", "204836_at", "204912_at", "204951_at",
   "204959_at", "205267_at", "205291_at", "205398_s_at",
   "205569_at", "205692_s_at", "205790_at", "206060_s_at",
   "207160_at", "207540_s_at", "207641_at", "207655_s_at",
   "207777_s_at", "207968_s_at", "208010_s_at", "209354_at",
   "209374_s_at", "209606_at", "209734_at", "209795_at",
   "212827_at", "AFFX-DapX-3_at", "AFFX-DapX-5_at",
   "AFFX-DapX-M_at", "AFFX-LysX-3_at", "AFFX-LysX-5_at",
   "AFFX-LysX-M_at", "AFFX-PheX-3_at", "AFFX-PheX-5_at",
   "AFFX-PheX-M_at", "AFFX-ThrX-3_at", "AFFX-ThrX-5_at",
   "AFFX-ThrX-M_at", "AFFX-r2-Bs-dap-3_at",
   "AFFX-r2-Bs-dap-5_at", "AFFX-r2-Bs-dap-M_at",
   "AFFX-r2-Bs-lys-3_at", "AFFX-r2-Bs-lys-5_at",
   "AFFX-r2-Bs-lys-M_at", "AFFX-r2-Bs-phe-3_at",
   "AFFX-r2-Bs-phe-5_at", "AFFX-r2-Bs-phe-M_at",
   "AFFX-r2-Bs-thr-3_s_at", "AFFX-r2-Bs-thr-5_s_at",
   "AFFX-r2-Bs-thr-M_s_at", "AFFX-r2-TagA_at",
   "AFFX-r2-TagB_at", "AFFX-r2-TagC_at", "AFFX-r2-TagD_at",
   "AFFX-r2-TagE_at", "AFFX-r2-TagF_at", "AFFX-r2-TagG_at",
   "AFFX-r2-TagH_at")

# These are the group numbers for each of the spiked-in
# clones given in the expanded.spike variable.  Positive
# numbers denote the original 42 clones reported by
# Affymetrix, negative numbers the supplemental 22 clones
# given by McGee et al., to facilitate separate analyses if
# necessary.
expanded.group <- c(8,10,1,2,6,9,1,1,3,4,10,2,6,3,7,4,5,8,8,
   5,9,9,2,4,10,-8,7,-5,6,7,3,5,13,-13,-13,14,-14,-14,14,-14,
   -14,14,-14,-14,-13,-13,-13,-14,-14,-14,-14,-14,-14,-14,-14,
   -14,11,11,11,12,12,12,13,13)
names(expanded.group) <- expanded.spike

# These are the categories (experiment numbers) to which
# each of the chips belongs
category <- rep(1:14,each=3)

# These are the categories (experiment numbers) for the
# comparisons.  By default, the baseline condition will be
# experiment 1.  All experiments in the list will be
# compared to the baseline in turn
cat.names <- unique(category[category!=1])

# This is a list of the filenames of the CEL files for this
# analysis. A separate variable is used to facilitate
```

```
# subanalyses if necessary.  For mmgmos, the model
# fitting is done over all chips.
small.fnames <- fnames
cel <- ReadAffy(filenames=small.fnames)
eset <- mmgmos(cel)
mmgmos.exprs <- exprs(eset)

# These are the number of probes in the expanded and
# original list of spiked-in clones.
num.large <- 64
num.small <- 42

# loop through the list of experiments for comparison
for (i in 1:length(cat.names)) {

# get the expression summary data for the comparison
    index <- category==1 | category==cat.names[i]
    data <- mmgmos.exprs[,index]

# construct the comparison vector for the multtest function
    compare <- c(rep(0,sum(as.integer(category==1))),rep(1,
       sum(as.integer(category==cat.names[i]))))
    t.df<-apply(data,1,df,compare=compare)
    ttest.mmgmos <- mt.teststat(data,compare)
    rawp0.mmgmos <- 2*(1-pt(abs(ttest.mmgmos),t.df))
    abs.score <- abs(rawp0.rma)
    gn <- geneNames(cel)
    index <- order(abs.score,decreasing=FALSE)

# rank the scores in decreasing order, with ties being
# assigned rank equal to the smallest rank of the group of
# ties
    ranking <- rank(abs.score,ties.method="min")

# reverse the rankings, as small p-values indicate higher
# probability of differential expression
    ranking <- max(ranking) - ranking + 1
    ranking <- rep(1:length(unique(ranking)),times=
       as.vector(table(ranking)))
    names(ranking) <- gn[index]

# create a data frame with the probeset (gene) names, rank,
# and score in order of increasing S-Scores
    results <- data.frame(name=gn[index],rank=ranking,score=
       abs.score[index])
```

```
    outfile <- paste("mmgmosFoldOverallU1133.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

# count the number of spike-in probes ranked in the top 42
# (for the original list) or top 64 (for the expanded list)
# of probesets
    small.count <- sum(!is.na(match(names(ranking)[1:
       num.small],spike.names)))
    large.count <- sum(!is.na(match(names(ranking)[1:
       num.large],expanded.spike)))
    results <- data.frame(small.count,large.count)
    outfile <- paste("mmgmosFoldCountU133.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

# create a data frame with the expected rank (based on fold
# change of the spike-in concentration) of the expanded list
# of spike-in probesets to compare to the actual rank (based
# on the S-Score values)
    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    outfile <- paste("mmgmosFoldRankU133.csv",sep="")
    results <- data.frame(name=expanded.spike,expectedrank=
       fold.rank[abs(expanded.group[expanded.spike])],
       actualrank=ranking[expanded.spike])
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

}

# End of Affymetrix U133 Analysis

#############################################################
#
# This performs an analysis of the Affymetrix U95 spike-in
# data set
#
# As this analysis is similar to the U133 analysis, only
# differences between the analyses will be highlighted.
#
#############################################################
```

```
fnames <- c("1521a99hpp_av06.CEL", "1521b99hpp_av06.CEL",
   "1521c99hpp_av06.CEL", "1521d99hpp_av06.CEL",
   "1521e99hpp_av06.CEL", "1521f99hpp_av06.CEL",
   "1521g99hpp_av06.CEL", "1521h99hpp_av06.CEL",
   "1521i99hpp_av06.CEL", "1521j99hpp_av06.CEL",
   "1521k99hpp_av06.CEL", "1521l99hpp_av06r.CEL",
   "1521m99hpp_av06.CEL", "1521n99hpp_av06.CEL",
   "1521o99hpp_av06.CEL", "1521p99hpp_av06.CEL",
   "1521q99hpp_av06.CEL", "1521r99hpp_av06.CEL",
   "1521s99hpp_av06.CEL", "1521t99hpp_av06.CEL",
   "1532a99hpp_av04.CEL", "1532b99hpp_av04.CEL",
   "1532c99hpp_av04.CEL", "1532d99hpp_av04.CEL",
   "1532e99hpp_av04.CEL", "1532f99hpp_av04.CEL",
   "1532g99hpp_av04.CEL", "1532h99hpp_av04.CEL",
   "1532i99hpp_av04.CEL", "1532j99hpp_av04.CEL",
   "1532k99hpp_av04.CEL", "1532l99hpp_av04.CEL",
   "1532m99hpp_av04.CEL", "1532n99hpp_av04.CEL",
   "1532o99hpp_av04.CEL", "1532p99hpp_av04.CEL",
   "1532q99hpp_av04.CEL", "1532r99hpp_av04.CEL",
   "1532s99hpp_av04.CEL", "1532t99hpp_av04r.CEL",
   "2353a99hpp_av08.CEL", "2353b99hpp_av08r.CEL",
   "2353d99hpp_av08.CEL", "2353e99hpp_av08.CEL",
   "2353f99hpp_av08.CEL", "2353g99hpp_av08.CEL",
   "2353h99hpp_av08.CEL", "2353i99hpp_av08.CEL",
   "2353j99hpp_av08.CEL", "2353k99hpp_av08.CEL",
   "2353l99hpp_av08.CEL", "2353m99hpp_av08.CEL",
   "2353n99hpp_av08.CEL", "2353o99hpp_av08.CEL",
   "2353p99hpp_av08.CEL", "2353q99hpp_av08.CEL",
   "2353r99hpp_av08.CEL", "2353s99hpp_av08.CEL",
   "2353t99hpp_av08.CEL")

spike.names <- c("37777_at", "684_at", "1597_at", "38734_at",
   "39058_at", "36311_at", "36889_at", "1024_at", "36202_at",
   "36085_at", "40322_at", "407_at", "1091_at", "1708_at")

spike.conc <- matrix(data=
   c(0,0.25,0.5,1,2,4,8,16,32,64,128,0,512,1024,
   0.25,0.5,1,2,4,8,16,32,64,128,256,0.25,1024,0,
   0.5,1,2,4,8,16,32,64,128,256,512,0.5,0,0.25,
   1,2,4,8,16,32,64,128,256,512,1024,1,0.25,0.5,
   2,4,8,16,32,64,128,256,512,1024,0,2,0.5,1,
   4,8,16,32,64,128,256,512,1024,0,0.25,4,1,2,
   8,16,32,64,128,256,512,1024,0,0.25,0.5,8,2,4,
   16,32,64,128,256,512,1024,0,0.25,0.5,1,16,4,8,
   32,64,128,256,512,1024,0,0.25,0.5,1,2,32,8,16,
```

```
   64,128,256,512,1024,0,0.25,0.5,1,2,4,64,16,32,
   128,256,512,1024,0,0.25,0.5,1,2,4,8,128,32,64,
   256,512,1024,0,0.25,0.5,1,2,4,8,16,256,64,128,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512), ncol=14,
   byrow=TRUE)

spike.group <- 1:14
names(spike.group) <- spike.names

# These are the categories (experiment numbers) for the
# comparisons.  Note that one chip in Experiment 3 of the
# U95 dataset did not hybridize properly, so that there are
# only 2 chips in this comparison rather than 3.  Though
# not originally intended, this allows the assessment of the
# algorithms when differing numbers of chips are compared.
category <- c(1:20,1:20,(1:20)[-3])
cat.names <- unique(category[category!=1])

small.fnames <- fnames
cel <- ReadAffy(filenames=small.fnames)
eset <- mmgmos(cel)
mmgmos.exprs <- exprs(eset)

num.large <- 14
num.small <- 14

for (i in 1:length(cat.names)) {

    data.mmgmos <- cbind(mmgmos.exprs[,category==1],
       mmgmos.exprs[,category==cat.names[i]])

    compare <- c(rep(0,sum(as.integer(category==1))),rep(1,
       sum(as.integer(category==cat.names[i]))))
    t.df<-apply(data.mmgmos,1,df,compare=compare)
    ttest.mmgmos <- mt.teststat(data.mmgmos,compare)
    rawp0.mmgmos <- 2*(1-pt(abs(ttest.mmgmos),t.df))
    abs.score <- abs(rawp0.mmgmos)
    gn <- geneNames(cel)
```

```
    index <- order(abs.score,decreasing=FALSE)

    ranking <- rank(abs.score,ties.method="min")
    ranking <- max(ranking) - ranking + 1
    ranking <- rep(1:length(unique(ranking)),times=
       as.vector(table(ranking)))
    names(ranking) <- gn[index]

    results <- data.frame(name=geneNames(cel)[index],
       iteration=rep(i,length(index)),rank=ranking,score=
       abs.score[index])
    outfile <- paste("mmgmosFoldOverallU95.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

    small.count <- sum(!is.na(match(names(ranking)[1:
       num.small],spike.names)))
    results <- data.frame(iteration=i,small.count)
    outfile <- paste("mmgmosFoldCountU95.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    results <- data.frame(name=spike.names,iteration=rep(i,
       length(spike.names)),expectedrank=
       fold.rank[spike.group[spike.names]],actualrank=
       ranking[spike.names])
    outfile <- paste("mmgmosFoldRankU95.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

}

# End of Affymetrix U95 analysis

############################################################
#
# This performs an analysis on the GeneLogic Dilution data
# set
#
# As this analysis is similar to the U133 and U95 analyses,
# only differences between the analyses will be highlighted.
```

```
#
############################################################

fnames <- c("92453hgu95a11.cel", "92454hgu95a11.cel",
   "92455hgu95a11.cel", "92456hgu95a11.cel",
   "92457hgu95a11.cel", "92458hgu95a11.cel",
   "92459hgu95a11.cel", "92460hgu95a11.cel",
   "92461hgu95a11.cel", "92462hgu95a11.cel",
   "92463hgu95a11.cel", "92464hgu95a11.cel",
   "92465hgu95a11.cel", "92466hgu95a11.cel",
   "92491hgu95a11.cel", "92492hgu95a11.cel",
   "92493hgu95a11.cel", "92494hgu95a11.cel",
   "92495hgu95a11.cel", "92496hgu95a11.cel",
   "92497hgu95a11.cel", "92498hgu95a11.cel",
   "92499hgu95a11.cel", "92500hgu95a11.cel",
   "92501hgu95a11.cel", "92503hgu95a11.cel")

fnames <- fnames[c(14,15,16,17,18,19,20,1,21,2,3,22,4,
   5,23,6,7,24,8,9,25,10,11,12,13,26)]

spike.conc <- matrix(data= c(0,0,0,0,0,0,0,0,0,0,
   0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,
   0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,
   1,1,1,1,1,1,1,1,1,1, 1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,
    2,2,2,2,2,2,2,2,2,2, 3,3,3,3,3,3,3,3,3,3,
   5,5,5,5,5,5,5,5,5,5,
   12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,
   25,25,25,25,25,25,25,25,25,25, 50,50,50,50,50,50,50,50,50,50,
    75,75,75,75,75,75,75,75,75,75,
   100,100,100,100,100,100,100,100,100,100,
   150,150,150,150,150,150,150,150,150,150), nrow=14,
   byrow=TRUE)

spike.names <- c("AFFX-BioB-5_at", "AFFX-BioB-M_at",
   "AFFX-BioB-3_at", "AFFX-BioC-5_at", "AFFX-BioC-3_at",
   "AFFX-BioDn-3_at", "AFFX-DapX-5_at", "AFFX-DapX-M_at",
   "AFFX-DapX-3_at", "AFFX-CreX-5_at")

spike.group <- 1:11
names(spike.group) <- spike.names

expanded.spike <- spike.names

expanded.group <- spike.group
names(expanded.group) <- expanded.spike
```

```
# These are the categories (experiment numbers) for the
# comparisons.  Note that only experiments 9 through 12 and
# experiment 14 have a sufficient number of chips for
# comparisons using all algorithms.  Thus, the baseline
# condition will be experiment 9.  All experiments in the
# list will be compared to the baseline in turn.
category <- c(1,2,3,4,5,6,7,8,8,rep(9:12,each=3),13,13,
   14,14,14)
cat.names <- unique(category[category > 9])

small.fnames <- fnames
cel <- ReadAffy(filenames=small.fnames)
eset <- mmgmos(cel)
mmgmos.exprs <- exprs(eset)

num.large <- 10
num.small <- 10

for (i in 1:length(cat.names)) {

    data.mmgmos <- cbind(mmgmos.exprs[,category==9],
       mmgmos.exprs[,category==cat.names[i]])

    compare <- c(rep(0,sum(as.integer(category==9))),rep(1,
       sum(as.integer(category==cat.names[i]))))
    t.df<-apply(data.mmgmos,1,df,compare=compare)
    ttest.mmgmos <- mt.teststat(data.mmgmos,compare)
    rawp0.mmgmos <- 2*(1-pt(abs(ttest.mmgmos),t.df))
    abs.score <- abs(rawp0.mmgmos)
    index <- order(abs.score,decreasing=TRUE)
    gn <- geneNames(cel)

    ranking <- rank(abs.score,ties.method="min")
    ranking <- max(ranking) - ranking + 1
    ranking <- rep(1:length(unique(ranking)),times=
       as.vector(table(ranking)))
    names(ranking) <- gn[index]
    results <- data.frame(name=gn[index],iteration=rep(i,
       length(index)),rank=ranking,score=abs.score[index])
    outfile <- paste("mmgmosFoldOverallGDilution.csv", sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

    posindex <- (abs.score <= 0.001)
```

```
    small.count <- sum(!is.na(match(gn[posindex], spike.names)))
    large.count <- sum(!is.na(match(gn[posindex],
       expanded.spike)))
    truepos <- large.count
    falsepos <- sum(posindex) - large.count
    falseneg <- num.large - large.count
    trueneg <- (length(abs.score) - sum(posindex)) - falseneg
    results <- data.frame(iteration=i,truepos,falsepos,
       trueneg,falseneg)
    outfile <- paste("mmgmosFoldCountGDilution.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    outfile <- paste("mmgmosFoldRankGDilution.csv",sep="")
    results <- data.frame(name=expanded.spike,iteration=rep(i,
       length(expanded.spike)),expectedrank=
       fold.rank[abs(expanded.group[expanded.spike])],
       actualrank=ranking[expanded.spike])
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

}
```

A.4 Multichip S-Score Analysis of Spike-In Datasets

```
#########################################################
#
# Program Name: MultiSScoreAnalysis.R
# Author: Richard Kennedy
# Date: 12/14/2007
#
# Purpose:  This program performs an automated analysis on
# several sets of data using the multichip SScore function
# as the primary analysis tool.
#
# Description: This program analyzes three separate
# datasets, the Affymetrix U95 and U133 Latin Square and the
# GeneLogic Dilution data.  For each dataset, the
# appropriate data files are read and the multichip S-Scores
# computed.  One data file is created showing the S-Scores
# for all of the probesets on the chip, in increasing order
# (or decreasing order of significance); one data file gives
# the number of spike-in probes (from both the original
# Affymetrix list and the expanded list of McGee et al.)
# that are highly ranked; and one data file shows the actual
# rank based on S-Scores versus the expected rank based on
# the concentration fold-change from the spike-in data.
# Although similar, separate computation routines are
# used for the Affymetrix U133 Latin Square, Affymetrix U95
# Latin Square, and GeneLogic Dilution datasets due to
# slight differences in the analyses and for better
# readability.
#
#########################################################

# Load the sscore library.  This is a standard library
# available through Bioconductor, which implements the
# multichip sscore function
library(sscore)

#########################################################
#
# This performs an analysis of the Affymetrix U133 spike-in
# data set
#
#########################################################
```

```
# These are the filenames, which are stored in order of the
# ASCII collating sequence, as in the directory listing.
fnames <- c("12_13_02_U133A_Mer_Latin_Square_Expt10_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt10_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt10_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt11_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt11_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt11_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt12_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt12_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt12_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt13_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt13_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt13_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt14_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt14_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt14_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt1_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt1_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt1_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt2_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt2_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt2_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt3_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt3_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt3_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt4_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt4_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt4_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt5_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt5_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt5_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt6_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt6_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt6_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt7_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt7_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt7_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt8_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt8_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt8_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt9_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt9_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt9_R3.CEL")
```

```
# Put the filenames in numerical order
fnames <- fnames[c(16:42,1:15)]

# These are the SF and SDT data, which were previously
# computed using the ComputeSFandSDT function in the sscore
# package.  By computing these beforehand, rather than on
# the fly, greatly speeds up the code.  These are stored in
# the same order as the original filenames, i.e. using the
# ASCII collating sequence from the directory listing
SF <- c(5.051527,6.725639,6.112969,5.55356,6.591798,5.515133,
   5.038274,5.457424,5.210437,5.213532,4.773091,5.170842,
   4.710175,5.675154,4.853165,4.231661,5.702423,4.860462,
   5.298943,4.589679,4.433209,5.372881,4.820332,4.853754,
   4.243081,4.640638,4.394098,4.354416,4.831838,4.471514,
   5.230509,6.470383,4.486733,4.493152,5.053464,5.441683,
   5.909456,5.4285,5.656854,5.397539,5.100904,5.460907)

# Put the SF data in the same order as the filenames data
SF <- SF[c(16:42,1:15)]

SDT <- c(19.13716,23.33307,22.39198,20.10130,26.19655,21.01957,
   18.94265,20.52486,18.65655,18.90886,17.14570,18.36193,
   16.54648,20.35357,17.8932,16.53504,20.99434,18.89438,
   20.23327,16.93843,16.72479,19.87543,17.24651,19.22224,
   17.82088,18.08628,16.85401,19.35906,20.43274,20.28572,
   19.90849,25.56167,17.58817,19.47506,22.84737,23.01953,
   24.11946,20.12949,22.82104,21.20344,19.66584,20.07275)
SDT <- SDT[c(16:42,1:15)]

# These are the names of the spiked-in clones for the
# original 42 probes reported by Affymetrix.  These are in
# the order given in the Affymetrix descriptor file included
# with the dataseta.  Note that there are 3 clones in each
# group of clones spiked in at the same concentration for a
# given experiment (see the Affymetrix descriptor file for
# additional information).
spike.names <- c("203508_at", "204563_at", "204513_s_at",
   "204205_at", "204959_at", "207655_s_at", "204836_at",
   "205291_at", "209795_at", "207777_s_at", "204912_at",
   "205569_at", "207160_at", "205692_s_at", "212827_at",
   "209606_at", "205267_at", "204417_at", "205398_s_at",
   "209734_at", "209354_at", "206060_s_at", "205790_at",
   "200665_s_at", "207641_at", "207540_s_at", "204430_s_at",
   "203471_s_at", "204951_at", "207968_s_at", "AFFX-r2-TagA_at",
    "AFFX-r2-TagB_at", "AFFX-r2-TagC_at", "AFFX-r2-TagD_at",
```

```
   "AFFX-r2-TagE_at", "AFFX-r2-TagF_at", "AFFX-r2-TagG_at",
   "AFFX-r2-TagH_at", "AFFX-DapX-3_at", "AFFX-LysX-3_at",
   "AFFX-PheX-3_at", "AFFX-ThrX-3_at")

# These are the concentration data for the clones in each
# experiment.  These are ordered across columns by clone
# group and across rows by experiment (or chip group).
spike.conc <- matrix(data=
   c(0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,
   0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,0,
   0.25,0.5,1,2,4,8,16,32,64,128,256,512,0,0.125,
   0.5,1,2,4,8,16,32,64,128,256,512,0,0.125,0.25,
   1,2,4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,
   2,4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,
   4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,
   8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,
   16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,
   32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,
   64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,
   128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,
   256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,
   512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256), nrow=14,
   byrow=TRUE)

# These are the group numbers for each of the spiked-in
# clones given in the spike.names variable
spike.group <- rep(1:14,each=3)
names(spike.group) <- spike.names

# These are the names of the spiked-in clones for the
# expanded set of 64 probes reported by McGee et al.  These
# are in the order given in their article and the
# supplemental files.  Note that there are no longer 3
# clones in each group when using the expanded set.
expanded.spike <- c("200665_s_at", "203471_s_at", "203508_at",
   "204205_at", "204417_at", "204430_s_at", "204513_s_at",
   "204563_at", "204836_at", "204912_at", "204951_at",
   "204959_at", "205267_at", "205291_at", "205398_s_at",
   "205569_at", "205692_s_at", "205790_at", "206060_s_at",
   "207160_at", "207540_s_at", "207641_at", "207655_s_at",
   "207777_s_at", "207968_s_at", "208010_s_at", "209354_at",
   "209374_s_at", "209606_at", "209734_at", "209795_at",
   "212827_at", "AFFX-DapX-3_at", "AFFX-DapX-5_at",
   "AFFX-DapX-M_at", "AFFX-LysX-3_at", "AFFX-LysX-5_at",
   "AFFX-LysX-M_at", "AFFX-PheX-3_at", "AFFX-PheX-5_at",
```

```
   "AFFX-PheX-M_at", "AFFX-ThrX-3_at", "AFFX-ThrX-5_at",
   "AFFX-ThrX-M_at", "AFFX-r2-Bs-dap-3_at",
   "AFFX-r2-Bs-dap-5_at", "AFFX-r2-Bs-dap-M_at",
   "AFFX-r2-Bs-lys-3_at", "AFFX-r2-Bs-lys-5_at",
   "AFFX-r2-Bs-lys-M_at", "AFFX-r2-Bs-phe-3_at",
   "AFFX-r2-Bs-phe-5_at", "AFFX-r2-Bs-phe-M_at",
   "AFFX-r2-Bs-thr-3_s_at", "AFFX-r2-Bs-thr-5_s_at",
   "AFFX-r2-Bs-thr-M_s_at", "AFFX-r2-TagA_at",
   "AFFX-r2-TagB_at", "AFFX-r2-TagC_at", "AFFX-r2-TagD_at",
   "AFFX-r2-TagE_at", "AFFX-r2-TagF_at", "AFFX-r2-TagG_at",
   "AFFX-r2-TagH_at")

# These are the group numbers for each of the spiked-in
# clones given in the expanded.spike variable.  Positive
# numbers denote the original 42 clones reported by
# Affymetrix, negative numbers the supplemental 22 clones
# given by McGee et al., to facilitate separate analyses if
# necessary.
expanded.group <- c(8,10,1,2,6,9,1,1,3,4,10,2,6,3,7,4,5,8,8,
   5,9,9,2,4,10,-8,7,-5,6,7,3,5,13,-13,-13,14,-14,-14,14,-14,
   -14,14,-14,-14,-13,-13,-13,-14,-14,-14,-14,-14,-14,-14,-14,
   -14,11,11,11,12,12,12,13,13)
names(expanded.group) <- expanded.spike

# These are the categories (experiment numbers) to which
# each of the chips belongs
category <- rep(1:14,each=3)

# These are the categories (experiment numbers) for the
# comparisons.  By default, the baseline condition will be
# experiment 1.  All experiments in the list will be
# compared to the baseline in turn
cat.names <- unique(category[category!=1])

# These are the number of probes in the expanded and
# original list of spiked-in clones.
num.large <- 64
num.small <- 42

# loop through the list of experiments for comparison
for (i in 1:length(cat.names)) {

# get the intensity data, SF, and SDT for the comparison
    index <- category==1 | category==cat.names[i]
```

```
# This is a list of the filenames of the CEL files for this
# analysis. A separate variable is used to facilitate
# subanalyses if necessary.
    small.fnames <- fnames[index]
    data <- ReadAffy(filenames=small.fnames)
    small.SF <- SF[index]
    small.SDT <- SDT[index]

# construct the comparison matrix for the sscore function
    compare <- c(rep(0,sum(as.integer(category==1))),rep(1,
       sum(as.integer(category==cat.names[i]))))

# compute the S-Scores
    score <- SScore(data,classlabel=compare,SF=small.SF,
       SDT=small.SDT)
    abs.score <- abs(exprs(score))
    index <- order(abs.score,decreasing=TRUE)
    gn <- geneNames(score)

# rank the scores in decreasing order, with ties being
# assigned rank equal to the smallest rank of the group of
# ties
    ranking <- rank(abs.score,ties.method="min")

# reverse the rankings, as small S-Scores indicate higher
# probability of differential expression
    ranking <- max(ranking) - ranking + 1
    ranking <- rep(1:length(unique(ranking)),times=
       as.vector(table(ranking)))
    names(ranking) <- gn[index]

# create a data frame with the probeset (gene) names, rank,
# and score in order of increasing S-Scores
    results <- data.frame(name=gn[index],iteration=rep(i,
       length(index)) ,rank=ranking,score=abs.score[index])
    outfile <- paste("SScoreFoldOverallU133.csv",sep="")
    write.table(results,file=outfile,sep=",",
       row.names=FALSE,col.names=(i==1),append=(i!=1))

# count the number of spike-in probes ranked in the top 42
# (for the original list) or top 64 (for the expanded list)
# of probesets
    small.count <- sum(!is.na(match(names(ranking)[1:
       num.small],spike.names)))
```

```
    large.count <- sum(!is.na(match(names(ranking)[1:
       num.large],expanded.spike)))
    results <- data.frame(iteration=i,small.count, large.count)
    outfile <- paste("SScoreFoldCountU133.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

# create a data frame with the expected rank (based on fold
# change of the spike-in concentration) of the expanded list
# of spike-in probesets to compare to the actual rank (based
# on the S-Score values)
    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    outfile <- paste("SScoreFoldRankU133.csv",sep="")
    results <- data.frame(name=expanded.spike,iteration=
       rep(i,length(expanded.spike)),expectedrank=
       fold.rank[abs(expanded.group[expanded.spike])],
       actualrank=ranking[expanded.spike])
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))
}

# End of Affymetrix U133 Analysis

##########################################################
#
# This performs an analysis of the Affymetrix U95 spike-in
# data set
#
# As this analysis is similar to the U133 analysis, only
# differences between the analyses will be highlighted.
#
##########################################################

fnames <- c("1521a99hpp_av06.CEL", "1521b99hpp_av06.CEL",
   "1521c99hpp_av06.CEL", "1521d99hpp_av06.CEL",
   "1521e99hpp_av06.CEL", "1521f99hpp_av06.CEL",
   "1521g99hpp_av06.CEL", "1521h99hpp_av06.CEL",
   "1521i99hpp_av06.CEL", "1521j99hpp_av06.CEL",
   "1521k99hpp_av06.CEL", "1521l99hpp_av06r.CEL",
   "1521m99hpp_av06.CEL", "1521n99hpp_av06.CEL",
   "1521o99hpp_av06.CEL", "1521p99hpp_av06.CEL",
   "1521q99hpp_av06.CEL", "1521r99hpp_av06.CEL",
```

```
   "1521s99hpp_av06.CEL", "1521t99hpp_av06.CEL",
   "1532a99hpp_av04.CEL", "1532b99hpp_av04.CEL",
   "1532c99hpp_av04.CEL", "1532d99hpp_av04.CEL",
   "1532e99hpp_av04.CEL", "1532f99hpp_av04.CEL",
   "1532g99hpp_av04.CEL", "1532h99hpp_av04.CEL",
   "1532i99hpp_av04.CEL", "1532j99hpp_av04.CEL",
   "1532k99hpp_av04.CEL", "1532l99hpp_av04.CEL",
   "1532m99hpp_av04.CEL", "1532n99hpp_av04.CEL",
   "1532o99hpp_av04.CEL", "1532p99hpp_av04.CEL",
   "1532q99hpp_av04.CEL", "1532r99hpp_av04.CEL",
   "1532s99hpp_av04.CEL", "1532t99hpp_av04r.CEL",
   "2353a99hpp_av08.CEL", "2353b99hpp_av08r.CEL",
   "2353d99hpp_av08.CEL", "2353e99hpp_av08.CEL",
   "2353f99hpp_av08.CEL", "2353g99hpp_av08.CEL",
   "2353h99hpp_av08.CEL", "2353i99hpp_av08.CEL",
   "2353j99hpp_av08.CEL", "2353k99hpp_av08.CEL",
   "2353l99hpp_av08.CEL", "2353m99hpp_av08.CEL",
   "2353n99hpp_av08.CEL", "2353o99hpp_av08.CEL",
   "2353p99hpp_av08.CEL", "2353q99hpp_av08.CEL",
   "2353r99hpp_av08.CEL", "2353s99hpp_av08.CEL",
   "2353t99hpp_av08.CEL")

SF <- c(15.54389,16.90462,18.58895,17.57569,18.10556,
   17.44596,19.05938,19.01886,15.19518,17.07320,17.70451,
   15.51397,15.14165,15.71093,18.92873,17.59843,17.74718,
   16.96576,19.88173,19.42636,14.22665,14.16075,11.39345,
   10.92027,15.86836,13.19469,15.66899,14.32828,11.25717,
   11.50788,13.16047,16.61321,13.50162,14.13247,12.45534,
   13.73491,13.59590,13.24143,14.58290,13.75632,16.37373,
   16.38092,13.26664,14.51221,15.94495,14.01617,13.29383,
   17.34152,12.74790,13.66622,17.21439,12.42648,13.16133,
   13.87641,18.97458,14.38793,14.25340,17.20059,15.56408)

SDT <- c(151.8202,164.8418,188.9549,179.0177,179.3972,
   169.2196,188.5424,185.0203,152.1097,198.8227,176.3480,
   210.3864,155.7234,163.5185,195.6335,187.2507,184.7659,
   186.7471,214.4948,204.1949,207.5883,190.5452,163.2673,
   158.7323,232.9045,188.8372,220.6606,205.0497,162.8883,
   181.9430,185.1602,224.2842,192.2052,206.1543,179.4869,
   202.1945,193.9658,196.5336,192.9138,136.5619,227.4531,
   205.0031,172.1437,179.7536,220.7491,181.4627,164.0342,
   203.9272,185.2207,168.6197,216.8669,164.1328,173.7930,
   185.0183,254.3695,177.5792,181.8116,215.0447,199.3295)
```

```
spike.names <- c("37777_at", "684_at", "1597_at", "38734_at",
   "39058_at", "36311_at", "36889_at", "1024_at", "36202_at",
   "36085_at", "40322_at", "407_at", "1091_at", "1708_at")

spike.conc <- matrix(data=
   c(0,0.25,0.5,1,2,4,8,16,32,64,128,0,512,1024,
   0.25,0.5,1,2,4,8,16,32,64,128,256,0.25,1024,0,
   0.5,1,2,4,8,16,32,64,128,256,512,0.5,0,0.25,
   1,2,4,8,16,32,64,128,256,512,1024,1,0.25,0.5,
   2,4,8,16,32,64,128,256,512,1024,0,2,0.5,1,
   4,8,16,32,64,128,256,512,1024,0,0.25,4,1,2,
   8,16,32,64,128,256,512,1024,0,0.25,0.5,8,2,4,
   16,32,64,128,256,512,1024,0,0.25,0.5,1,16,4,8,
   32,64,128,256,512,1024,0,0.25,0.5,1,2,32,8,16,
   64,128,256,512,1024,0,0.25,0.5,1,2,4,64,16,32,
   128,256,512,1024,0,0.25,0.5,1,2,4,8,128,32,64,
   256,512,1024,0,0.25,0.5,1,2,4,8,16,256,64,128,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512), ncol=14,
   byrow=TRUE)

spike.group <- 1:14
names(spike.group) <- spike.names

# These are the categories (experiment numbers) for the
# comparisons.  Note that one chip in Experiment 3 of the
# U95 dataset did not hybridize properly, so that there are
# only 2 chips in this comparison rather than 3.  Though not
# originally intended, this allows the assessment of the
# algorithms when differing numbers of chips are compared.
category <- c(1:20,1:20,(1:20)[-3])
cat.names <- unique(category[category!=1])

num.large <- 14
num.small <- 14

for (i in 1:length(cat.names)) {
```

```
small.fnames <- c(fnames[category==1],fnames[category==
   cat.names[i]])
data <- ReadAffy(filenames=small.fnames)
small.SF <- c(SF[category==1],SF[category== cat.names[i]])
small.SDT <- c(SDT[category==1],SDT[category==
   cat.names[i]])

compare <- c(rep(0,sum(as.integer(category==1))),rep(1,
   sum(as.integer(category==cat.names[i]))))
score <- SScore(data,classlabel=compare,SF=small.SF,SDT=
   small.SDT)
abs.score <- abs(exprs(score))
gn <- geneNames(score)

index <- order(abs.score,decreasing=TRUE)
ranking <- rank(abs.score,ties.method="min")
ranking <- max(ranking) - ranking + 1
ranking <- rep(1:length(unique(ranking)),times=
   as.vector(table(ranking)))
names(ranking) <- gn[index]

results <- data.frame(name=gn[index],iteration=rep(i,
   length(index)),rank=ranking,score=abs.score[index])
outfile <- paste("SScoreFoldOverallU95.csv",sep="")
write.table(results,file=outfile,sep=",",row.names=
   FALSE,col.names=(i==1),append=(i!=1))

small.count <- sum(!is.na(match(names(ranking)[1:
   num.small],spike.names)))
results <- data.frame(iteration=i,small.count)
outfile <- paste("SScoreFoldCountU95.csv",sep="")
write.table(results,file=outfile,sep=",",row.names=
   FALSE,col.names=(i==1),append=(i!=1))

fold.change <- spike.conc[i+1,] / spike.conc[1,]
fold.rank <- rank(fold.change,ties.method="min")
fold.rank <- rep(length(unique(fold.rank)):1,times=
   as.vector(table(fold.rank)))
results <- data.frame(name=spike.names,iteration=rep(i,
   length(spike.names)),expectedrank=
   fold.rank[spike.group[spike.names]],actualrank=
   ranking[spike.names])
outfile <- paste("SScoreFoldRankU95.csv",sep="")
write.table(results,file=outfile,sep=",",row.names=
   FALSE,col.names=(i==1),append=(i!=1))
```

```
}

# End of Affymetrix U95 analysis

############################################################
#
# This performs an analysis on the GeneLogic Dilution
# dataset.
#
# As this analysis is similar to the U133 and U95 analyses,
# only differences between the analyses will be highlighted.
#
############################################################

fnames <- c("92453hgu95a11.cel", "92454hgu95a11.cel",
   "92455hgu95a11.cel", "92456hgu95a11.cel",
   "92457hgu95a11.cel", "92458hgu95a11.cel",
   "92459hgu95a11.cel", "92460hgu95a11.cel",
   "92461hgu95a11.cel", "92462hgu95a11.cel",
   "92463hgu95a11.cel", "92464hgu95a11.cel",
   "92465hgu95a11.cel", "92466hgu95a11.cel",
   "92491hgu95a11.cel", "92492hgu95a11.cel",
   "92493hgu95a11.cel", "92494hgu95a11.cel",
   "92495hgu95a11.cel", "92496hgu95a11.cel",
   "92497hgu95a11.cel", "92498hgu95a11.cel",
   "92499hgu95a11.cel", "92500hgu95a11.cel",
   "92501hgu95a11.cel", "92503hgu95a11.cel")

SF <- c(12.870930,9.969553,10.633744,5.498765,5.835703,
   7.732482,11.598326,10.133451,6.454634,5.355627,7.001940,
   8.849713,7.280378,12.280841,7.331615,19.023698,7.380893,
   18.712581,7.834392,6.895325,7.254859,21.076266,10.342030,
   7.940419,15.479335,14.88527)

SDT <- c(133.89655,101.82649,114.87093,37.85469,48.64600,
   86.43693,143.22880,111.54653,48.30689,31.88864,49.17773,
   91.11646,43.77653,122.86836,52.00780,172.02050,50.62660,
   169.93095,46.71437,48.85516,56.43905,191.73461,66.32453,
   53.81287,146.23999,144.8691)

fnames <- fnames[c(14,15,16,17,18,19,20,1,21,2,3,22,4,5,23,
   6,7,24,8,9,25,10,11,12,13,26)]

spike.conc <- matrix(data= c(0,0,0,0,0,0,0,0,0,0,
   0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,
```

```
   0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,
   1,1,1,1,1,1,1,1,1,1, 1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,
    2,2,2,2,2,2,2,2,2,2, 3,3,3,3,3,3,3,3,3,3,
   5,5,5,5,5,5,5,5,5,5,
   12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,
   25,25,25,25,25,25,25,25,25,25, 50,50,50,50,50,50,50,50,50,50,
    75,75,75,75,75,75,75,75,75,75,
   100,100,100,100,100,100,100,100,100,100,
   150,150,150,150,150,150,150,150,150,150), nrow=14,
   byrow=TRUE)

spike.names <- c("AFFX-BioB-5_at", "AFFX-BioB-M_at",
   "AFFX-BioB-3_at", "AFFX-BioC-5_at", "AFFX-BioC-3_at",
   "AFFX-BioDn-3_at", "AFFX-DapX-5_at", "AFFX-DapX-M_at",
   "AFFX-DapX-3_at", "AFFX-CreX-5_at")

spike.group <- 1:11
names(spike.group) <- spike.names

# These are the categories (experiment numbers) for the
# comparisons.  Note that only experiments 9 through 12 and
# experiment 14 have a sufficient number of chips for
# comparisons using all algorithms.  Thus, the baseline
# condition will be experiment 9.  All experiments in the
# list will be compared to the baseline in turn.
category <- c(1,2,3,4,5,6,7,8,8,rep(9:12,each=3),13,13,14,
   14,14)
cat.names <- unique(category[category > 9 & category != 13])

num.large <- 10
num.small <- 10

for (i in 1:length(cat.names)) {

    index <- category==9 | category==cat.names[i]
    small.fnames <- fnames[index]
    data <- ReadAffy(filenames=small.fnames)
    small.SF <- SF[index]
    small.SDT <- SDT[index]

    compare <- c(rep(0,sum(as.integer(category==9))),rep(1,
       sum(as.integer(category==cat.names[i]))))

    score <- SScore(data,classlabel=compare,SF=small.SF,SDT=
       small.SDT)
```

```
    abs.score <- abs(exprs(score))
    gn <- geneNames(score)

    index <- order(abs.score,decreasing=TRUE)
    ranking <- rank(abs.score,ties.method="min")
    ranking <- max(ranking) - ranking + 1
    ranking <- rep(1:length(unique(ranking)),times=
       as.vector(table(ranking)))
    names(ranking) <- gn[index]

    results <- data.frame(name=gn[index],iteration=rep(i,
       length(index)),rank=ranking,score=abs.score[index])
    outfile <- paste("SScoreFoldOverallGDilution.csv", sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

    posindex <- (abs.score >= 3.29)
    small.count <- sum(!is.na(match(gn[posindex],spike.names)))
    truepos <- large.count
    falsepos <- sum(posindex) - large.count
    falseneg <- num.large - large.count
    trueneg <- (length(abs.score) - sum(posindex)) - falseneg
    results <- data.frame(iteration=i,truepos,falsepos,
       trueneg,falseneg)
    outfile <- paste("SScoreFoldCountGDilution.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    outfile <- paste("SScoreFoldRankGDilution.csv",sep="")
    results <- data.frame(name=spike.names,iteration=rep(i,
       length(spike.names)),expectedrank=
       fold.rank[spike.group[spike.names]],actualrank=
       ranking[spike.names])
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))
}

# End of GeneLogic Dilution Analysis
```

A.5 Pooled S-Score Analysis of Spike-In Datasets

```
#########################################################
#
# Program Name: PooledSScoreAnalysis.R
# Author: Richard Kennedy
# Date: 12/21/2007
#
# Purpose:  This program performs an automated analysis on
# several sets of data using the pooled S-Score algorithm as #
   the expression summary measure.
#
# Description: This program analyzes three separate
# datasets, the Affymetrix U95 and U133 Latin Square and the
# GeneLogic Dilution data.  For each dataset, the
# appropriate data files are read and the pooled S-Scores
# are computed.  One data file is created showing the pooled #
   S-Scores for all of the probesets on the chip, in
# increasing order (or decreasing order of significance);
# one data file gives the number of spike-in probes (from
# both the original Affymetrix list and the expanded list of
# McGee et al.) that are highly ranked; and one data file
# shows the actual rank based on S-Scores versus the
# expected rank based on the concentration fold-change from
# the spike-in data.  Although similar, separate computation #
   routines are used for the Affymetrix U133 Latin Square,
# Affymetrix U95 Latin Square, and GeneLogic Dilution
# datasets due to slight differences in the analyses and for
# better readability.
#
#########################################################

# Load the sscore library.  This is a standard library
# available through Bioconductor, which implements the
# multichip sscore function
library(sscore)

# Load the robust library.  This is a standard library
# available through CRAN, which implements the lmRob
# function for robust regression
library(robust)

# This function implements the pooled S-Score.  The code
# is similar to the Bioconductor package sscore, but the
```

```
# error estimate is calculated using a modification of the
# local pooled error (LPE) method of Jain et al.  Code for
# the error estimate is adapted from the LPE package, also
# available through Bioconductor.
# Input:
# Output:
PooledSScore <- function(afbatch = stop("No CEL files
   specified"),
 conditions =  stop("No list of comparisons given"), SF =
    NULL,SDT =
 NULL, rm.outliers = TRUE,rm.mask = TRUE, rm.extra = TRUE,
    digits =
 NULL,verbose = FALSE,celfile.path = NULL, celfile.names =
    NULL,quant=0.001) {

    fname <- sampleNames(afbatch)

# Identify outliers using the computeOutlier function from
# the sscore package.  This returns a matrix with rows
# corresponding to probe and columns to chip.  The value
# of an element in the matrix is set to TRUE if the probe
# on that chip is an outlier, otherwise it is set to FALSE.
    outlier <- computeOutlier(afbatch)

# Initialize various variables used in the computations.  l1
# and l2 represent vectors of the variances for the
# probesets in the baseline and experimental conditions,
# respectively.  Variances are assumed to be the same for
# each probeset across chips, so that only one variance
# per probeset is computed.  pnames is a vector of probe
# names in the same order as l1 and l2.
    l1 <- l2 <- NULL
    pnames <- NULL

# This is the gamma proportionality constant from the
# original S-Score article.
    m.gamma <- 0.1

# Score is a vector of S-Scores, one for each probeset.
# probenames stores the probeset names in the same order as
# Score.
    probenames <- geneNames(afbatch)
    Score <- CorrDiff <- rep(0.0,length(probenames))
    writeLines("Computing S-score values")
```

```
# Get the indices for the the PM and MM probes for lookup
    pmidx <- pmindex(afbatch)
    mmidx <- mmindex(afbatch)

# Get the intensity values for each condition, with intens1
# being the baseline and intens2 being the experimental
# condition
    intens1 <- t(t(intensity(afbatch[,conditions==0]))*
       SF[conditions==0])
    intens2 <- t(t(intensity(afbatch[,conditions==1]))*
       SF[conditions==1])

# Find the maximum intensity for each condition
    max1 <- apply(rbind(pm(afbatch[,conditions==0]),
       mm(afbatch[,conditions==0])),2,max)*SF[conditions==0]
    max2 <- apply(rbind(pm(afbatch[,conditions==1]),
       mm(afbatch[,conditions==1])),2,max)*SF[conditions==1]

# this loops through each of the probesets on a given pair
# of chips
    for (i in 1:length(probenames)) {

# get the PM and MM values, as well as minimum intensity
# values, for the given probeset on each chip of the pair
        PM1 <- intens1[pmidx[[i]],,drop=FALSE]
        MM1 <- intens1[mmidx[[i]],,drop=FALSE]
        PM2 <- intens2[pmidx[[i]],,drop=FALSE]
        MM2 <- intens2[mmidx[[i]],,drop=FALSE]
        min1 <- apply(rbind(PM1,MM1),2,min)
        min2 <- apply(rbind(PM2,MM2),2,min)

# adjust each of the PM and MM intensities relative to the
# minimum values
        PM1 <- t(t(PM1) - min1)
        PM2 <- t(t(PM2) - min2)
        MM1 <- t(t(MM1) - min1)
        MM2 <- t(t(MM2) - min2)

# find the index of the probe pairs of the probeset to use
# in calculations.  A probe pair is used if it is not
# "saturated" (i.e., the intensity is less than the maximum
# - minimum) and if it is not identified as an outlier /
# masked value in the .CEL file
        index <- cbind((PM1<max1-min1),(PM2<max2-min2),(MM1<
           max1-min1),(MM2<max2-min2))
```

```
        index <- apply(index,1,any)
        if (any(rm.outliers,rm.mask,rm.extra)) {
            outlier1 <- outlier[pmidx[[i]],conditions==0,
               drop=FALSE] | outlier[mmidx[[i]],conditions==0,
               drop=FALSE]
            outlier2 <- outlier[pmidx[[i]],conditions==1,
               drop=FALSE] | outlier[mmidx[[i]],conditions==1,
               drop=FALSE]
            index <- index & (!apply(outlier1,1,any)) &
               (!apply(outlier2,1,any))
        }
        N <- sum(as.integer(index))

# find the PM-MM differences for the probeset on each of the
# two chips in the pair
        diff1 <- (PM1-MM1)[index,,drop=FALSE]
        diff2 <- (PM2-MM2)[index,,drop=FALSE]

# Append the PM-MM differences for this probeset to the
# running list
        l1 <- rbind(l1,diff1)
        l2 <- rbind(l2,diff2)

    }

# Using the PM-MM differences for each condition, calculate
# the variance using the LPE method.
# First find the mean intensity for each row (probeset)
# across chips for the baseline condition
    l1.means <- rowMeans(l1)

# Find the quantiles for the intensities.  For each probeset
# having a mean intensity within a quantile, assign all
# intensity values for that probeset (across chips) to the
# group for that quantile
    quantile.l1 <- quantile(l1.means, probs = seq(0, 1, quant),
       na.rm = TRUE)
    index <- rep(0,length(l1.means)-1)
    for (i in 2:length(quantile.l1)) {
        index[l1.means > quantile.l1[i-1] & l1.means <=
           quantile.l1[i]] <- i-1
    }

# Find the mean and variance (using trimmed mean for the
```

```
# former and the median absolute deviation for the latter)  #
   for each quantile, if it exists.  Note that it is possible
# for consecutive quantiles to have the same threshold
# points, which is particularly likely in the low intensity
# region where many probesets have the same intensity value.
# In this case, all of the probesets are assigned to the
# group for the "first" quantile of the consecutive set,
# leaving the "remaining" quantiles of the consecutive set
# empty.  Assign these "remaining" quantiles to have NA
# mean and variance.
    mean.l1 <- var.l1 <- rep(0,length(quantile.l1)-1)
    for (i in 1:(length(quantile.l1)-1)) {
        if (length(l1[index==i,]) != 0) {
            mean.l1[i] <- mean(as.vector(l1[index==i,]),
               na.rm=TRUE,trim=0.125)
            var.l1[i] <- mad(as.vector(l1[index==i,]),
               na.rm=TRUE)
        } else {
            mean.l1[i] <- NA
            var.l1[i] <- NA
        }
    }

# Now address the quantiles having NA mean and variance,
# using an approach based on the original LPE method.  If
# the mean and variance of the "previous" quantile is not
# NA, then the quantile having NA mean and variance is
# assigned a mean and variance that is the average of the
# "previous" and "next" quantiles.  Otherwise the quantile
# having NA mean and variance is assigned the mean and
# variance of the "next" quantile, which should not be NA at
# the uppermost intensities because these are sparsely
# populated.

    if (any(is.na(var.l1)) ) {
        for (i in (length(var.l1)-1):1 ) {
            if (is.na(var.l1[i])) {
                  var.l1[i] <- ifelse(!is.na(var.l1[i-1]),
                     mean(var.l1[i+1], var.l1[i-1]),
                     var.l1[i+1])
            }
        }
    }

# Perform a robust regression of variance on mean squared
```

```
# intensity to estimate the fitting parameters
    mean.squared <- as.vector(mean.l1)^2
    lm.l1 <- lmRob(as.vector(var.l1) ~ mean.squared)

# Compute the predicted variance for each probeset based on
# the robust regression model, which will be used in the
# calculation of the S-Score values
    alpha <- lm.l1$coefficients[1]
    gamma <- lm.l1$coefficients[2]
    predict.var1 <- gamma*l1^2+alpha

# Perform a similar computation of variance for the
# experimental condition
    l2.means <- rowMeans(l2)
    quantile.l2 <- quantile(l2.means, probs = seq(0, 1, quant),
       na.rm = TRUE)
    index <- rep(0,length(l2.means)-1)
    for (i in 2:length(quantile.l2)) {
        index[l2.means > quantile.l2[i-1] & l2.means <=
           quantile.l2[i]] <- i-1
    }
    mean.l2 <- var.l2 <- rep(0,length(quantile.l2)-1)
    for (i in 1:(length(quantile.l2)-1)) {
        if (length(l2[index==i,]) != 0) {
            mean.l2[i] <- mean(as.vector(l2[index==i,]),
               na.rm=TRUE,trim=0.125)
            var.l2[i] <- mad(as.vector(l2[index==i,]),
               na.rm=TRUE)
        } else {
            mean.l2[i] <- NA
            var.l2[i] <- NA
        }
    }

    if (any(is.na(var.l2)) ) {
        for (i in (length(var.l2)-1):1 ) {
            if (is.na(var.l2[i])) {
                  var.l2[i] <- ifelse(!is.na(var.l2[i-1]),

                     mean(var.l2[i+1], var.l2[i-1]),

                     var.l2[i+1])
            }
        }
    }
```

```
    mean.squared <- as.vector(mean.l2)^2
    lm.l2 <- lmRob(as.vector(var.l2) ~ mean.squared)
    alpha <- lm.l2$coefficients[1]
    gamma <- lm.l2$coefficients[2]
    predict.var2 <- gamma*l2^2+alpha

# Note that the PM-MM differences are not associated with
# their respective probesets for calculation of the S-Score
# values.  Thus, some of the S-Score calculations must be
# performed again, since the variance estimates were not
# available previously.  This inefficiency will be corrected
# in later versions.
    for (i in 1:length(probenames)) {
        PM1 <- intens1[pmidx[[i]],,drop=FALSE]
        MM1 <- intens1[mmidx[[i]],,drop=FALSE]
        PM2 <- intens2[pmidx[[i]],,drop=FALSE]
        MM2 <- intens2[mmidx[[i]],,drop=FALSE]
        min1 <- apply(rbind(PM1,MM1),2,min)
        min2 <- apply(rbind(PM2,MM2),2,min)

# adjust each of the PM and MM intensities relative to the
# minimum values
        PM1 <- t(t(PM1) - min1)
        PM2 <- t(t(PM2) - min2)
        MM1 <- t(t(MM1) - min1)
        MM2 <- t(t(MM2) - min2)

# find the index of the probe pairs of the probeset to use
# in calculations.  A probe pair is used if it is not
# "saturated" (i.e., the intensity is less than the maximum
# - minimum) and if it is not identified as an outlier /
# masked value in the .CEL file
        index <- cbind((PM1<max1-min1),(PM2<max2-min2),(MM1<
           max1-min1),(MM2<max2-min2))
        index <- apply(index,1,any)
        if (any(rm.outliers,rm.mask,rm.extra)) {
            outlier1 <- outlier[pmidx[[i]],conditions==0,
               drop=FALSE] | outlier[mmidx[[i]],conditions==0,
               drop=FALSE]
            outlier2 <- outlier[pmidx[[i]],conditions==1,
               drop=FALSE] | outlier[mmidx[[i]],conditions==1,
               drop=FALSE]
```

```
            index <- index & (!apply(outlier1,1,any)) &
               (!apply(outlier2,1,any))
        }
        N <- sum(as.integer(index))

# find the PM-MM differences for the probeset on each of the
# two chips in the pair
        diff1 <- (PM1-MM1)[index,,drop=FALSE]
        diff2 <- (PM2-MM2)[index,,drop=FALSE]

# Compute the S-Score values.  This follows the formula in
# the original program (and the J Mol Biol article) except
# the predicted variance based on the LPE method is
# substituted for the variance based on the SDT value
        f.err <- (apply(diff1,1,sum)-apply(diff2,1,sum))/
           sqrt(sum(predict.var1[i,]) + sum(predict.var2[i,]))

# threshold or impute outlying f.err values.  The cutoff of
# 15 was arbitrarily decided in the original version; how it
# was determined is unknown
        f.err[f.err > 15.0] <- 15.0
        f.err[f.err < -15.0] <- -15.0

        Score[i] <- sum(f.err)

# estimate the variance / covariance values, for calculating
# the CorrDiff
        Sxx <- sum(mean(diff1)^2)
        Syy <- sum(mean(diff2)^2)
        Sxy <- sum(mean(diff1)*mean(diff2))

        Sx <- 0
        Sy <- 0

# transform the S-Score estimate by dividing by a function
# of the number of probes in the probeset
        if (N > 0)
            Score[i] <- Score[i] / sqrt(N) else
            Score[i] <- 0

# calculate the CorrDiff.  CorrDiffs below the threshold of
# 1e-3 are imputed to be 0, which was also arbitrarily
# decided in the first version
        if (N>2 && ((Sxx-Sx*Sx/N)*(Syy-Sy*Sy/N) > 1.e-3))
```

```
            CorrDiff[i] <- (Sxy-Sx*Sy/N)/sqrt((Sxx-Sx*Sx/N)*
               (Syy-Sy*Sy/N)) else
            CorrDiff[i] <- 0.0

    }

# now renormalize the S-Score values, which gives alpha
    writeLines("Renormalizing S-scores")
    x <- Score
    Sx <- sum(x)
    Sxx <- sum(x*x)

# calculate the mean and standard deviation of the entire
# set of S-Scores
    Sstdev <- sqrt((Sxx-Sx*Sx/length(Score))/length(Score))
    meanSx <- Sx/length(Score)

# find the trimmed S-score values, using a cutoff of those
# S-Scores within 3 standard deviations of the mean
    x <- Score-meanSx;
    x <- x[abs(x) < 3*Sstdev]
    Sx <- sum(x)
    Sxx <- sum(x*x)
    num <- length(x)

# calculate the trimmed mean and standard deviation.  Again,
# the cutoff of 0.01 was arbitrarily decided in the first
# version
    Sstdev <- ((Sxx-Sx*Sx/num)/num)
    if (Sstdev < 0.01)
        Sstdev <- 1.0 else
        Sstdev <- sqrt(Sstdev)
    m.alpha <- Sstdev
    meanSx <- Sx/num+meanSx

# perform the renormalization, using the trimmed mean and
# standard deviation values
    Score <- (Score-meanSx)/Sstdev

    fn1 <- (fname[conditions==0])[1]
    fn2 <- (fname[conditions==1])[1]

# output information on these parameters if desired by the
# user
    if (verbose) {
```

```
        Chip <- cdfName(afbatch)
        num.probesets <- length(Score)
        writeLines("S-score data. Parameter section:")
        writeLines(sprintf("Probearray type:      %s", Chip))
        writeLines(sprintf("sample1:      %s", fn1))
        writeLines(sprintf("sample2:      %s", fn2))
        writeLines(sprintf("Alpha--error coupling factor within
           a probeset:     %8.3f",m.alpha))
        writeLines(sprintf("Gamma--weight of multiplicative
           error:     %8.3f",m.gamma))
        writeLines(sprintf("Number of Probesets:
           %i",num.probesets))
        writeLines(" ")
        writeLines("Scaling Factor:")
        printSF <- formatC(SF[conditions==0],digits=3,
           width=8,format="f")
        writeLines(sprintf("  sample1 (class label 0):
           %s",paste(printSF,collapse=" ")))
        printSF <- formatC(SF[conditions==1],digits=3,
           width=8,format="f")
        writeLines(sprintf("  sample2 (class label 1):
           %s",paste(printSF,collapse=" ")))
        writeLines("SDT background noise:")
        printSDT <- formatC(SDT[conditions==0],digits=3,
           width=8,format="f")
        writeLines(sprintf("  sample1 (class label 0):
           %s",paste(printSDT,collapse=" ")))
        printSDT <- formatC(SDT[conditions==1],digits=3,
           width=8,format="f")
        writeLines(sprintf("  sample2 (class label 1):
           %s",paste(printSDT,collapse=" ")))
        writeLines("Max Intensity:")
        printMax <- formatC(max1,digits=3,width=8, format="f")
        writeLines(sprintf("  sample1 (class label 0):
           %s",paste(printMax,collapse=" ")))
        printMax <- formatC(max2,digits=3,width=8, format="f")
        writeLines(sprintf("  sample2 (class label 1):
           %s",paste(printMax,collapse=" ")))
        writeLines(" ")
    }

# round the S-Scores and CorrDiff to the number of digits
# specified by the user.  For the desktop version, this was
# 3
    if (!is.null(digits)) {
```

```
        Score <- round(Score,digits)
        CorrDiff <- round(CorrDiff,digits)
    }

    Score <- as.matrix(Score)
    CorrDiff <- as.matrix(CorrDiff)
    rownames(Score) <- rownames(CorrDiff) <- geneNames(afbatch)
    colnames(Score) <- colnames(CorrDiff) <- "Class 0 vs 1"
    comparison <- 1
    Score.pData <- data.frame(comparison,row.names="Class 0 vs
       1")
    Score.Metadata <- data.frame(labelDescription = "arbitrary
       numbering",
      row.names = "comparison")
    ScorePheno <- new("AnnotatedDataFrame", data=Score.pData,
       varMetadata =
      Score.Metadata)
# put the values into an ExprSet to return.  The phenoData,
# annotation, and description are the same as the AffyBatch
# object
    eset <- new("ExpressionSet",
        exprs=Score,
        phenoData=ScorePheno,
        annotation=annotation(afbatch))
    return(eset)
}

## end of adapted code

############################################################
#
# This performs an analysis of the Affymetrix U133 spike-in
# data set
#
############################################################

# These are the filenames, which are stored in order of the
# ASCII collating sequence, as in the directory listing.
fnames <- c("12_13_02_U133A_Mer_Latin_Square_Expt10_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt10_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt10_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt11_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt11_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt11_R3.CEL",
```

```
"12_13_02_U133A_Mer_Latin_Square_Expt12_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt12_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt12_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt13_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt13_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt13_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt14_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt14_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt14_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt1_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt1_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt1_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt2_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt2_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt2_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt3_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt3_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt3_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt4_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt4_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt4_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt5_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt5_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt5_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt6_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt6_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt6_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt7_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt7_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt7_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt8_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt8_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt8_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt9_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt9_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt9_R3.CEL")

# Put the filenames in numerical order
fnames <- fnames[c(16:42,1:15)]

# These are the SF and SDT data, which were previously
# computed using the ComputeSFandSDT function in the sscore
# package.  By computing these beforehand, rather than on
# the fly, greatly speeds up the code.  These are stored in
# the same order as the original filenames, i.e. using the
```

```
# ASCII collating sequence from the directory listing
SF <- c(5.051527, 6.725639, 6.112969, 5.55356, 6.591798,
   5.515133, 5.038274, 5.457424, 5.210437, 5.213532, 4.773091,
   5.170842, 4.710175, 5.675154, 4.853165, 4.231661, 5.702423,
   4.860462, 5.298943, 4.589679, 4.433209, 5.372881, 4.820332,
   4.853754, 4.243081, 4.640638, 4.394098, 4.354416, 4.831838,
   4.471514, 5.230509, 6.470383, 4.486733, 4.493152, 5.053464,
   5.441683, 5.909456, 5.4285, 5.656854, 5.397539, 5.100904,
   5.460907)

# Put the SF data in the same order as the filenames data
SF <- SF[c(16:42,1:15)]

SDT <- c(19.13716, 23.33307, 22.39198, 20.10130, 26.19655,
   21.01957, 18.94265, 20.52486, 18.65655, 18.90886, 17.14570,
   18.36193, 16.54648, 20.35357, 17.8932, 16.53504, 20.99434,
   18.89438, 20.23327, 16.93843, 16.72479, 19.87543, 17.24651,
   19.22224, 17.82088, 18.08628, 16.85401, 19.35906, 20.43274,
   20.28572, 19.90849, 25.56167, 17.58817, 19.47506, 22.84737,
   23.01953, 24.11946, 20.12949, 22.82104, 21.20344, 19.66584,
   20.07275)
SDT <- SDT[c(16:42,1:15)]

# These are the names of the spiked-in clones for the
# original 42 probes reported by Affymetrix.  These are in
# the order given in the Affymetrix descriptor file included
# with the dataseta.  Note that there are 3 clones in each
# group of clones spiked in at the same concentration for a
# given experiment (see the Affymetrix descriptor file for
# additional information).
spike.names <- c("203508_at", "204563_at", "204513_s_at",
   "204205_at", "204959_at", "207655_s_at", "204836_at",
   "205291_at", "209795_at", "207777_s_at", "204912_at",
   "205569_at", "207160_at", "205692_s_at", "212827_at",
   "209606_at", "205267_at", "204417_at", "205398_s_at",
   "209734_at", "209354_at", "206060_s_at", "205790_at",
   "200665_s_at", "207641_at", "207540_s_at", "204430_s_at",
   "203471_s_at", "204951_at", "207968_s_at", "AFFX-r2-TagA_at",
    "AFFX-r2-TagB_at", "AFFX-r2-TagC_at", "AFFX-r2-TagD_at",
   "AFFX-r2-TagE_at", "AFFX-r2-TagF_at", "AFFX-r2-TagG_at",
   "AFFX-r2-TagH_at", "AFFX-DapX-3_at", "AFFX-LysX-3_at",
   "AFFX-PheX-3_at", "AFFX-ThrX-3_at")

# These are the concentration data for the clones in each
# experiment.  These are ordered across columns by clone
```

```
# group and across rows by experiment (or chip group).
spike.conc <- spike.conc <- matrix(data=
   c(0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,
   0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,0,
   0.25,0.5,1,2,4,8,16,32,64,128,256,512,0,0.125,
   0.5,1,2,4,8,16,32,64,128,256,512,0,0.125,0.25,
   1,2,4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,
   2,4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,
   4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,
   8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,
   16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,
   32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,
   64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,
   128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,
   256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,
   512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256), nrow=14,
   byrow=TRUE)

# These are the group numbers for each of the spiked-in
# clones given in the spike.names variable
spike.group <- rep(1:14,each=3)
names(spike.group) <- spike.names

# These are the names of the spiked-in clones for the
# expanded set of 64 probes reported by McGee et al.  These
# are in the order given in their article and the
# supplemental files.  Note that there are no longer 3
# clones in each group when using the expanded set.
expanded.spike <- c("200665_s_at", "203471_s_at", "203508_at",
   "204205_at", "204417_at", "204430_s_at", "204513_s_at",
   "204563_at", "204836_at", "204912_at", "204951_at",
   "204959_at", "205267_at", "205291_at", "205398_s_at",
   "205569_at", "205692_s_at", "205790_at", "206060_s_at",
   "207160_at", "207540_s_at", "207641_at", "207655_s_at",
   "207777_s_at", "207968_s_at", "208010_s_at", "209354_at",
   "209374_s_at", "209606_at", "209734_at", "209795_at",
   "212827_at", "AFFX-DapX-3_at", "AFFX-DapX-5_at",
   "AFFX-DapX-M_at", "AFFX-LysX-3_at", "AFFX-LysX-5_at",
   "AFFX-LysX-M_at", "AFFX-PheX-3_at", "AFFX-PheX-5_at",
   "AFFX-PheX-M_at", "AFFX-ThrX-3_at", "AFFX-ThrX-5_at",
   "AFFX-ThrX-M_at", "AFFX-r2-Bs-dap-3_at",
   "AFFX-r2-Bs-dap-5_at", "AFFX-r2-Bs-dap-M_at",
   "AFFX-r2-Bs-lys-3_at", "AFFX-r2-Bs-lys-5_at",
   "AFFX-r2-Bs-lys-M_at", "AFFX-r2-Bs-phe-3_at",
   "AFFX-r2-Bs-phe-5_at", "AFFX-r2-Bs-phe-M_at",
```

```
   "AFFX-r2-Bs-thr-3_s_at", "AFFX-r2-Bs-thr-5_s_at",
   "AFFX-r2-Bs-thr-M_s_at", "AFFX-r2-TagA_at",
   "AFFX-r2-TagB_at", "AFFX-r2-TagC_at", "AFFX-r2-TagD_at",
   "AFFX-r2-TagE_at", "AFFX-r2-TagF_at", "AFFX-r2-TagG_at",
   "AFFX-r2-TagH_at")

# These are the group numbers for each of the spiked-in
# clones given in the expanded.spike variable.  Positive
# numbers denote the original 42 clones reported by
# Affymetrix, negative numbers the supplemental 22 clones
# given by McGee et al., to facilitate separate analyses if
# necessary.
expanded.group <- c(8,10,1,2,6,9,1,1,3,4,10,2,6,3,7,4,5,8,8,
   5,9,9,2,4,10,-8,7,-5,6,7,3,5,13,-13,-13,14,-14,-14,14,-14,
   -14,14,-14,-14,-13,-13,-13,-14,-14,-14,-14,-14,-14,-14,-14,
   -14,11,11,11,12,12,12,13,13)
names(expanded.group) <- expanded.spike

# These are the categories (experiment numbers) to which
# each of the chips belongs
category <- rep(1:14,each=3)

# These are the categories (experiment numbers) for the
# comparisons.  By default, the baseline condition will be
# experiment 1.  All experiments in the list will be
# compared to the baseline in turn
cat.names <- unique(category[category!=1])

# These are the number of probes in the expanded and
# original list of spiked-in clones.
num.large <- 64
num.small <- 42

# loop through the list of experiments for comparison
for (i in 1:length(cat.names)) {

# get the intensity data, SF, and SDT for the comparison
    index <- category==1 | category==cat.names[i]

# This is a list of the filenames of the CEL files for this
# analysis. A separate variable is used to facilitate
# subanalyses if necessary.
    small.fnames <- fnames[index]
    data <- ReadAffy(filenames=small.fnames)
    small.SF <- SF[index]
```

```
    small.SDT <- SDT[index]

# construct the comparison matrix for the sscore function
    compare <- c(rep(0,sum(as.integer(category==1))),rep(1,
       sum(as.integer(category==cat.names[i]))))

# compute the S-Scores
    score <- PooledSScore(data,conditions=compare,SF=
       small.SF,SDT=small.SDT)
    abs.score <- abs(exprs(score))
    index <- order(abs.score,decreasing=TRUE)
    gn <- geneNames(score)

# rank the scores in decreasing order, with ties being
# assigned rank equal to the smallest rank of the group of
# ties
    ranking <- rank(abs.score,ties.method="min")

# reverse the rankings, as small S-Scores indicate higher
# probability of differential expression
    ranking <- max(ranking) - ranking + 1
    ranking <- rep(1:length(unique(ranking)),times=
       as.vector(table(ranking)))
    names(ranking) <- gn[index]

# create a data frame with the probeset (gene) names, rank,
# and score in order of increasing S-Scores
    results <- data.frame(name=gn[index],
       iteration=rep(i,length(index)),rank=ranking,score=
       abs.score[index])
    outfile <- paste("PooledSScoreFoldOverallU133.csv", sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

# count the number of spike-in probes ranked in the top 42
# (for the original list) or top 64 (for the expanded list)
# of probesets
    small.count <- sum(!is.na(match(names(ranking)[1:
       num.small],spike.names)))
    large.count <- sum(!is.na(match(names(ranking)[1:
       num.large], expanded.spike)))
    results <- data.frame(iteration=i,small.count, large.count)
    outfile <- paste("PooledSScoreFoldCountU133.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))
```

```
# create a data frame with the expected rank (based on fold
# change of the spike-in concentration) of the expanded list
# of spike-in probesets to compare to the actual rank (based
# on the S-Score values)
    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    outfile <- paste("PooledSScoreFoldRankU133.csv",sep="")
    results <- data.frame(name=expanded.spike,iteration=
       rep(i,length(expanded.spike)),expectedrank=
       fold.rank[abs(expanded.group[expanded.spike])],
       actualrank=ranking[expanded.spike])
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

}

# end of Affymetrix U133A analysis

############################################################
#
# This performs an analysis of the Affymetrix U95 spike-in
# data set
#
# As this analysis is similar to the U133 analysis, only
# differences between the analyses will be highlighted.
#
############################################################

fnames <- c("1521a99hpp_av06.CEL", "1521b99hpp_av06.CEL",
   "1521c99hpp_av06.CEL", "1521d99hpp_av06.CEL",
   "1521e99hpp_av06.CEL", "1521f99hpp_av06.CEL",
   "1521g99hpp_av06.CEL", "1521h99hpp_av06.CEL",
   "1521i99hpp_av06.CEL", "1521j99hpp_av06.CEL",
   "1521k99hpp_av06.CEL", "1521l99hpp_av06r.CEL",
   "1521m99hpp_av06.CEL", "1521n99hpp_av06.CEL",
   "1521o99hpp_av06.CEL", "1521p99hpp_av06.CEL",
   "1521q99hpp_av06.CEL", "1521r99hpp_av06.CEL",
   "1521s99hpp_av06.CEL", "1521t99hpp_av06.CEL",
   "1532a99hpp_av04.CEL", "1532b99hpp_av04.CEL",
   "1532c99hpp_av04.CEL", "1532d99hpp_av04.CEL",
   "1532e99hpp_av04.CEL", "1532f99hpp_av04.CEL",
   "1532g99hpp_av04.CEL", "1532h99hpp_av04.CEL",
```

```
   "1532i99hpp_av04.CEL", "1532j99hpp_av04.CEL",
   "1532k99hpp_av04.CEL", "1532l99hpp_av04.CEL",
   "1532m99hpp_av04.CEL", "1532n99hpp_av04.CEL",
   "1532o99hpp_av04.CEL", "1532p99hpp_av04.CEL",
   "1532q99hpp_av04.CEL", "1532r99hpp_av04.CEL",
   "1532s99hpp_av04.CEL", "1532t99hpp_av04r.CEL",
   "2353a99hpp_av08.CEL", "2353b99hpp_av08r.CEL",
   "2353d99hpp_av08.CEL", "2353e99hpp_av08.CEL",
   "2353f99hpp_av08.CEL", "2353g99hpp_av08.CEL",
   "2353h99hpp_av08.CEL", "2353i99hpp_av08.CEL",
   "2353j99hpp_av08.CEL", "2353k99hpp_av08.CEL",
   "2353l99hpp_av08.CEL", "2353m99hpp_av08.CEL",
   "2353n99hpp_av08.CEL", "2353o99hpp_av08.CEL",
   "2353p99hpp_av08.CEL", "2353q99hpp_av08.CEL",
   "2353r99hpp_av08.CEL", "2353s99hpp_av08.CEL",
   "2353t99hpp_av08.CEL")

SF <- c(15.54389, 16.90462, 18.58895, 17.57569, 18.10556,
   17.44596, 19.05938, 19.01886, 15.19518, 17.07320, 17.70451,
   15.51397, 15.14165, 15.71093, 18.92873, 17.59843, 17.74718,
   16.96576, 19.88173, 19.42636, 14.22665, 14.16075, 11.39345,
   10.92027, 15.86836, 13.19469, 15.66899, 14.32828, 11.25717,
   11.50788, 13.16047, 16.61321, 13.50162, 14.13247, 12.45534,
   13.73491, 13.59590, 13.24143, 14.58290, 13.75632, 16.37373,
   16.38092, 13.26664, 14.51221, 15.94495, 14.01617, 13.29383,
   17.34152, 12.74790, 13.66622, 17.21439, 12.42648, 13.16133,
   13.87641, 18.97458, 14.38793, 14.25340, 17.20059, 15.56408)

SDT <- c(151.8202, 164.8418, 188.9549, 179.0177, 179.3972,
   169.2196, 188.5424, 185.0203, 152.1097, 198.8227, 176.3480,
   210.3864, 155.7234, 163.5185, 195.6335, 187.2507, 184.7659,
   186.7471, 214.4948, 204.1949, 207.5883, 190.5452, 163.2673,
   158.7323, 232.9045, 188.8372, 220.6606, 205.0497, 162.8883,
   181.9430, 185.1602, 224.2842, 192.2052, 206.1543, 179.4869,
   202.1945, 193.9658, 196.5336, 192.9138, 136.5619, 227.4531,
   205.0031, 172.1437, 179.7536, 220.7491, 181.4627, 164.0342,
   203.9272, 185.2207, 168.6197, 216.8669, 164.1328, 173.7930,
   185.0183, 254.3695, 177.5792, 181.8116, 215.0447, 199.3295)

spike.names <- c("37777_at", "684_at", "1597_at", "38734_at",
   "39058_at", "36311_at", "36889_at", "1024_at", "36202_at",
   "36085_at", "40322_at", "407_at", "1091_at", "1708_at")

spike.conc <- matrix(data=
   c(0,0.25,0.5,1,2,4,8,16,32,64,128,0,512,1024,
```

```
   0.25,0.5,1,2,4,8,16,32,64,128,256,0.25,1024,0,
   0.5,1,2,4,8,16,32,64,128,256,512,0.5,0,0.25,
   1,2,4,8,16,32,64,128,256,512,1024,1,0.25,0.5,
   2,4,8,16,32,64,128,256,512,1024,0,2,0.5,1,
   4,8,16,32,64,128,256,512,1024,0,0.25,4,1,2,
   8,16,32,64,128,256,512,1024,0,0.25,0.5,8,2,4,
   16,32,64,128,256,512,1024,0,0.25,0.5,1,16,4,8,
   32,64,128,256,512,1024,0,0.25,0.5,1,2,32,8,16,
   64,128,256,512,1024,0,0.25,0.5,1,2,4,64,16,32,
   128,256,512,1024,0,0.25,0.5,1,2,4,8,128,32,64,
   256,512,1024,0,0.25,0.5,1,2,4,8,16,256,64,128,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512),  ncol=14,
   byrow=TRUE)

spike.group <- 1:14
names(spike.group) <- spike.names

small.fnames <- fnames

# These are the categories (experiment numbers) for the
# comparisons.  Note that one chip in Experiment 3 of the
# U95 dataset did not hybridize properly, so that there are
# only 2 chips in this comparison rather than 3.  Though not
# originally intended, this allows the assessment of the
# algorithms when differing numbers of chips are compared.
category <- c(1:20,1:20,(1:20)[-3])
cat.names <- unique(category[category!=1])

num.large <- 14
num.small <- 14

for (i in 1:length(cat.names)) {

    small.fnames <- c(fnames[category==1],fnames[category==
       cat.names[i]])
    data <- ReadAffy(filenames=small.fnames)
    small.SF <- c(SF[category==1],SF[category== cat.names[i]])
```

```
small.SDT <- c(SDT[category==1],SDT[category==
   cat.names[i]])

compare <- c(rep(0,sum(as.integer(category==1))),rep(1,
   sum(as.integer(category==cat.names[i]))))

score <- PooledSScore(data,conditions=compare,SF=
   small.SF,SDT=small.SDT)
abs.score <- abs(exprs(score))
index <- order(abs.score,decreasing=TRUE)
gn <- geneNames(score)

ranking <- rank(abs.score,ties.method="min")

ranking <- max(ranking) - ranking + 1
ranking <- rep(1:length(unique(ranking)),times=
   as.vector(table(ranking)))
names(ranking) <- gn[index]

results <- data.frame(name=gn[index],iteration=
   rep(i,length(index)),rank=ranking,score=
   abs.score[index])
outfile <- paste("PooledSScoreFoldOverallU95.csv", sep="")
write.table(results,file=outfile,sep=",",row.names=
   FALSE,col.names=(i==1),append=(i!=1))

small.count <- sum(!is.na(match(names(ranking)[1:
   num.small],spike.names)))
results <- data.frame(iteration=i,small.count)
outfile <- paste("PooledSScoreFoldCountU95.csv",sep="")
write.table(results,file=outfile,sep=",",row.names=
   FALSE,col.names=(i==1),append=(i!=1))

fold.change <- spike.conc[i+1,] / spike.conc[1,]
fold.rank <- rank(fold.change,ties.method="min")
fold.rank <- rep(length(unique(fold.rank)):1,times=
   as.vector(table(fold.rank)))
results <- data.frame(name=spike.names,iteration=rep(i,
   length(spike.names)),expectedrank=
   fold.rank[spike.group[spike.names]],actualrank=
   ranking[spike.names])
outfile <- paste("PooledSScoreFoldRankU95.csv",sep="")
write.table(results,file=outfile,sep=",",row.names=
   FALSE,col.names= (i==1),append=(i!=1))
```

```
}

# end of Affymetrix U95 analysis


#############################################################
#
# This performs an analysis on the GeneLogic Dilution data
# set
#
# As this analysis is similar to the U133 and U95 analyses,
# only differences between the analyses will be highlighted.
#
#############################################################

fnames <- c("92453hgu95a11.cel", "92454hgu95a11.cel",
   "92455hgu95a11.cel", "92456hgu95a11.cel",
   "92457hgu95a11.cel", "92458hgu95a11.cel",
   "92459hgu95a11.cel", "92460hgu95a11.cel",
   "92461hgu95a11.cel", "92462hgu95a11.cel",
   "92463hgu95a11.cel", "92464hgu95a11.cel",
   "92465hgu95a11.cel", "92466hgu95a11.cel",
   "92491hgu95a11.cel", "92492hgu95a11.cel",
   "92493hgu95a11.cel", "92494hgu95a11.cel",
   "92495hgu95a11.cel", "92496hgu95a11.cel",
   "92497hgu95a11.cel", "92498hgu95a11.cel",
   "92499hgu95a11.cel", "92500hgu95a11.cel",
   "92501hgu95a11.cel", "92503hgu95a11.cel")

SF <- c(12.870930, 9.969553, 10.633744, 5.498765, 5.835703,
   7.732482, 11.598326, 10.133451, 6.454634, 5.355627, 7.001940,
    8.849713, 7.280378, 12.280841, 7.331615, 19.023698,
   7.380893, 18.712581, 7.834392, 6.895325, 7.254859, 21.076266,
    10.342030, 7.940419, 15.479335, 14.88527)

SDT <- c(133.89655, 101.82649, 114.87093, 37.85469, 48.64600,
   86.43693, 143.22880, 111.54653, 48.30689, 31.88864, 49.17773,
    91.11646, 43.77653, 122.86836, 52.00780, 172.02050,
   50.62660, 169.93095, 46.71437, 48.85516, 56.43905, 191.73461,
    66.32453, 53.81287, 146.23999, 144.8691)

fnames <- fnames[c(14,15,16,17,18,19,20,1,21,2,3,22,4,5,23,
   6,7,24,8,9,25,10,11,12,13,26)]
```

```
spike.conc <- matrix(data= c(0,0,0,0,0,0,0,0,0,0,
   0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,
   0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,
   1,1,1,1,1,1,1,1,1,1, 1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,
    2,2,2,2,2,2,2,2,2,2, 3,3,3,3,3,3,3,3,3,3,
   5,5,5,5,5,5,5,5,5,5,
   12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,
   25,25,25,25,25,25,25,25,25,25, 50,50,50,50,50,50,50,50,50,50,
    75,75,75,75,75,75,75,75,75,75,
   100,100,100,100,100,100,100,100,100,100,
   150,150,150,150,150,150,150,150,150,150), nrow=14,
   byrow=TRUE)

spike.names <- c("AFFX-BioB-5_at", "AFFX-BioB-M_at",
   "AFFX-BioB-3_at", "AFFX-BioC-5_at", "AFFX-BioC-3_at",
   "AFFX-BioDn-3_at", "AFFX-DapX-5_at", "AFFX-DapX-M_at",
   "AFFX-DapX-3_at", "AFFX-CreX-5_at")

spike.group <- 1:11
names(spike.group) <- spike.names

expanded.spike <- spike.names

expanded.group <- spike.group
names(expanded.group) <- expanded.spike

# These are the categories (experiment numbers) for the
# comparisons.  Note that only experiments 9 through 12 and
# experiment 14 have a sufficient number of chips for
# comparisons using all algorithms.  Thus, the baseline
# condition will be experiment 9.  All experiments in the
# list will be compared to the baseline in turn.
category <- c(1,2,3,4,5,6,7,8,8,rep(9:12,each=3),13,13,14,14,14)
cat.names <- unique(category[category > 9])

num.large <- 10
num.small <- 10

for (i in 1:length(cat.names)) {

    index <- category==9 | category==cat.names[i]
    small.fnames <- fnames[index]
    data <- ReadAffy(filenames=small.fnames)
    small.SF <- SF[index]
    small.SDT <- SDT[index]
```

```
compare <- c(rep(0,sum(as.integer(category==9))),rep(1,
   sum(as.integer(category==cat.names[i]))))

score <- PooledSScore(data,conditions=compare,SF=
   small.SF,SDT=small.SDT)
abs.score <- abs(exprs(score))
index <- order(abs.score,decreasing=TRUE)
gn <- geneNames(score)

ranking <- rank(abs.score,ties.method="min")
ranking <- max(ranking) - ranking + 1
ranking <- rep(1:length(unique(ranking)),times=
   as.vector(table(ranking)))
names(ranking) <- gn[index]

results <- data.frame(name=gn[index],iteration=
   rep(i,length(index)),rank=ranking,score=
   abs.score[index])
outfile <- paste("PooledSScoreFoldOverallGDilution.csv",
   sep="")
write.table(results,file=outfile,sep=",",row.names=
   FALSE,col.names= (i==1),append=(i!=1))

posindex <- (abs.score >= 3.29)
small.count <- sum(!is.na(match(gn[posindex], spike.names)))
large.count <- sum(!is.na(match(gn[posindex],
   expanded.spike)))
truepos <- large.count
falsepos <- sum(posindex) - large.count
falseneg <- num.large - large.count
trueneg <- (length(abs.score) - sum(posindex)) -  falseneg
results <- data.frame(iteration=i,truepos,falsepos,trueneg,
   falseneg)
outfile <- paste("PooledSScoreFoldCountGDilution.csv",
   sep="")
write.table(results,file=outfile,sep=",",row.names=
   FALSE,col.names=(i==1),append=(i!=1))

fold.change <- spike.conc[i+1,] / spike.conc[1,]
fold.rank <- rank(fold.change,ties.method="min")
fold.rank <- rep(length(unique(fold.rank)):1,times=
   as.vector(table(fold.rank)))
outfile <- paste("PooledSScoreFoldRankGDilution.csv",
   sep="")
```

```
    results <- data.frame(name=expanded.spike,iteration=
       rep(i,length(expanded.spike)),expectedrank=
       fold.rank[abs(expanded.group[expanded.spike])],
       actualrank=ranking[expanded.spike])
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

}

# End of GeneLogic Dilution Analysis
```

A.6 RMA Analysis of Spike-In Datasets

```
#############################################################
#
# Program Name: RMAAnalysis.R
# Author: Richard Kennedy
# Date: 12/21/2007
#
# Purpose:  This program performs an automated analysis on
# several sets of data using the RMA algorithm as the
# expression summary measure.
#
# Description: This program analyzes three separate
# datasets, the Affymetrix U95 and U133 Latin Square and the
# GeneLogic Dilution data.  For each dataset, the
# appropriate data files are read and the RMA expression
# summary computed.  The RMA values are then compared using
# multiple t-tests to give measures of significance for
# differential gene expression.  One data file is created
# showing the p-values of the t-tests for all of the
# probesets on the chip, in increasing order (or decreasing
# order of significance); one data file gives the number of
# spike-in probes (from both the original Affymetrix list
# and the expanded list of McGee et al.) that are highly
# ranked; and one data file shows the actual rank based on
# the p-values versus the expected rank based on the
# concentration fold-change from the spike-in data.
# Although similar, separate computation routines are used
# for the Affymetrix U133 Latin Square, Affymetrix U95 Latin
# Square, and GeneLogic Dilution datasets due to slight
# differences in the analyses and for better readability.
#
#############################################################

# Load the affy library.  This is a standard library
# available through Bioconductor, which implements the
# functions for reading CEL files
library(affy)

# Load the multtest library.  This is a standard library
# available through Bioconductor, which implements the
# multiple t-test among other functions.
library(multtest)
```

```
# This function implements a pooled degrees of freedom
# function, which computes a composite degrees of freedom
# for a two-sample comparison based on the relative size
# of each of the two samples.
# Input: exprs - a matrix containing the expression values
#        compare - a vector denoting the condition of each
#                  column in the exprs matrix, with 0 denoting
#                  the baseline condition and 1 denoting the
#                  experimental condition
# Output: a vector containing the pooled degrees of freedom
#         for each row of the exprs matrix
df <- function(exprs,compare) {
    var1 <- var(exprs[compare==1])
    var0 <- var(exprs[compare==0])
    n1 <- length(exprs[compare==1])
    n0 <- length(exprs[compare==0])
    result <- (var1+var0)^2 / (var1^2/(n1-1)+var0^2/(n0-1))
    return(result)
}

# End of declared functions

############################################################
#
# This performs an analysis of the Affymetrix U133 spike-in
# data set
#
############################################################

# These are the filenames, which are stored in order of the
# ASCII collating sequence, as in the directory listing.
fnames <- c("12_13_02_U133A_Mer_Latin_Square_Expt10_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt10_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt10_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt11_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt11_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt11_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt12_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt12_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt12_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt13_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt13_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt13_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt14_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt14_R2.CEL",
```

```
   "12_13_02_U133A_Mer_Latin_Square_Expt14_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt1_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt1_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt1_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt2_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt2_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt2_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt3_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt3_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt3_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt4_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt4_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt4_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt5_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt5_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt5_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt6_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt6_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt6_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt7_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt7_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt7_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt8_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt8_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt8_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt9_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt9_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt9_R3.CEL")

# Put the filenames in numerical order
fnames <- fnames[c(16:42,1:15)]

# These are the names of the spiked-in clones for the
# original 42 probes reported by Affymetrix.  These are in
# the order given in the Affymetrix descriptor file included
# with the dataseta.  Note that there are 3 clones in each
# group of clones spiked in at the same concentration for a
# given experiment (see the Affymetrix descriptor file for
# additional information).
spike.names <- c("203508_at", "204563_at", "204513_s_at",
   "204205_at", "204959_at", "207655_s_at", "204836_at",
   "205291_at", "209795_at", "207777_s_at", "204912_at",
   "205569_at", "207160_at", "205692_s_at", "212827_at",
   "209606_at", "205267_at", "204417_at", "205398_s_at",
   "209734_at", "209354_at", "206060_s_at", "205790_at",
```

```
   "200665_s_at", "207641_at", "207540_s_at", "204430_s_at",
   "203471_s_at", "204951_at", "207968_s_at", "AFFX-r2-TagA_at",
    "AFFX-r2-TagB_at", "AFFX-r2-TagC_at", "AFFX-r2-TagD_at",
   "AFFX-r2-TagE_at", "AFFX-r2-TagF_at", "AFFX-r2-TagG_at",
   "AFFX-r2-TagH_at", "AFFX-DapX-3_at", "AFFX-LysX-3_at",
   "AFFX-PheX-3_at", "AFFX-ThrX-3_at")

# These are the concentration data for the clones in each
# experiment.  These are ordered across columns by clone
# group and across rows by experiment (or chip group).
spike.conc <- matrix(data=
   c(0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,
   0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,0,
   0.25,0.5,1,2,4,8,16,32,64,128,256,512,0,0.125,
   0.5,1,2,4,8,16,32,64,128,256,512,0,0.125,0.25,
   1,2,4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,
   2,4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,
   4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,
   8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,
   16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,
   32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,
   64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,
   128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,
   256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,
   512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256), nrow=14,
   byrow=TRUE)

# These are the group numbers for each of the spiked-in
# clones given in the spike.names variable
spike.group <- rep(1:14,each=3)
names(spike.group) <- spike.names

# These are the names of the spiked-in clones for the
# expanded set of 64 probes reported by McGee et al.  These
# are in the order given in their article and the
# supplemental files.  Note that there are no longer 3
# clones in each group when using the expanded set.
expanded.spike <- c("200665_s_at", "203471_s_at", "203508_at",
   "204205_at", "204417_at", "204430_s_at", "204513_s_at",
   "204563_at", "204836_at", "204912_at", "204951_at",
   "204959_at", "205267_at", "205291_at", "205398_s_at",
   "205569_at", "205692_s_at", "205790_at", "206060_s_at",
   "207160_at", "207540_s_at", "207641_at", "207655_s_at",
   "207777_s_at", "207968_s_at", "208010_s_at", "209354_at",
   "209374_s_at", "209606_at", "209734_at", "209795_at",
```

```
   "212827_at", "AFFX-DapX-3_at", "AFFX-DapX-5_at",
   "AFFX-DapX-M_at", "AFFX-LysX-3_at", "AFFX-LysX-5_at",
   "AFFX-LysX-M_at", "AFFX-PheX-3_at", "AFFX-PheX-5_at",
   "AFFX-PheX-M_at", "AFFX-ThrX-3_at", "AFFX-ThrX-5_at",
   "AFFX-ThrX-M_at", "AFFX-r2-Bs-dap-3_at",
   "AFFX-r2-Bs-dap-5_at", "AFFX-r2-Bs-dap-M_at",
   "AFFX-r2-Bs-lys-3_at", "AFFX-r2-Bs-lys-5_at",
   "AFFX-r2-Bs-lys-M_at", "AFFX-r2-Bs-phe-3_at",
   "AFFX-r2-Bs-phe-5_at", "AFFX-r2-Bs-phe-M_at",
   "AFFX-r2-Bs-thr-3_s_at", "AFFX-r2-Bs-thr-5_s_at",
   "AFFX-r2-Bs-thr-M_s_at", "AFFX-r2-TagA_at",
   "AFFX-r2-TagB_at", "AFFX-r2-TagC_at", "AFFX-r2-TagD_at",
   "AFFX-r2-TagE_at", "AFFX-r2-TagF_at", "AFFX-r2-TagG_at",
   "AFFX-r2-TagH_at")

# These are the group numbers for each of the spiked-in
# clones given in the expanded.spike variable.  Positive
# numbers denote the original 42 clones reported by
# Affymetrix, negative numbers the supplemental 22 clones
# given by McGee et al., to facilitate separate analyses if
# necessary.
expanded.group <- c(8,10,1,2,6,9,1,1,3,4,10,2,6,3,7,4,5,8,8,
   5,9,9,2,4,10,-8,7,-5,6,7,3,5,13,-13,-13,14,-14,-14,14,-14,
   -14,14,-14,-14,-13,-13,-13,-14,-14,-14,-14,-14,-14,-14,-14,
   -14,11,11,11,12,12,12,13,13)
names(expanded.group) <- expanded.spike

# These are the categories (experiment numbers) to which
# each of the chips belongs
category <- rep(1:14,each=3)

# These are the categories (experiment numbers) for the
# comparisons.  By default, the baseline condition will be
# experiment 1.  All experiments in the list will be
# compared to the baseline in turn
cat.names <- unique(category[category!=1])

# This is a list of the filenames of the CEL files for this
# analysis. A separate variable is used to facilitate
# subanalyses if necessary.  For RMA, the normalization will
# be done over all chips, as this seems to be the commonly
# accepted practice.
small.fnames <- fnames
cel <- ReadAffy(filenames=small.fnames)
eset <- rma(cel)
```

```
rma.exprs <- exprs(eset)

# These are the number of probes in the expanded and
# original list of spiked-in clones.
num.large <- 64
num.small <- 42

# loop through the list of experiments for comparison
for (i in 1:length(cat.names)) {

# get the expression summary data for the comparison
    index <- category==1 | category==cat.names[i]
    data <- rma.exprs[,index]

# construct the comparison vector for the multtest function
    compare <- c(rep(0,sum(as.integer(category==1))),rep(1,
       sum(as.integer(category==cat.names[i]))))
    t.df<-apply(data,1,df,compare=compare)
    ttest.rma <- mt.teststat(data,compare)
    rawp0.rma <- 2*(1-pt(abs(ttest.rma),t.df))
    abs.score <- abs(rawp0.rma)
    gn <- geneNames(cel)
    index <- order(abs.score,decreasing=FALSE)

# rank the scores in decreasing order, with ties being
# assigned rank equal to the smallest rank of the group of
# ties
    ranking <- rank(abs.score,ties.method="min")

# reverse the rankings, as small p-values indicate higher
# probability of differential expression
    ranking <- max(ranking) - ranking + 1
    ranking <- rep(1:length(unique(ranking)),times=
       as.vector(table(ranking)))
    names(ranking) <- gn[index]

# create a data frame with the probeset (gene) names, rank,
# and score in order of increasing S-Scores
    results <- data.frame(name=geneNames(cel)[index],rank=
       ranking, score=abs.score[index])
    outfile <- paste("RMAFoldOverallU133.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))
    num.zero <- sum(abs.score==0)
```

```
# count the number of spike-in probes ranked in the top 42
# (for the original list) or top 64 (for the expanded list)
# of probesets
    small.count <- sum(!is.na(match(names(ranking)[1:
       max(num.small, num.zero)],spike.names)))
    large.count <- sum(!is.na(match(names(ranking)[1:
       max(num.large, num.zero)],expanded.spike)))
    results <- data.frame(small.count,large.count)
    outfile <- paste("RMAFoldCountU133.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

# create a data frame with the expected rank (based on fold
# change of the spike-in concentration) of the expanded list
# of spike-in probesets to compare to the actual rank (based
# on the S-Score values)
    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    outfile <- paste("RMAFoldRankU133.csv",sep="")
    results <- data.frame(name=expanded.spike, expectedrank=
       fold.rank[abs(expanded.group[expanded.spike])],
       actualrank=ranking[expanded.spike])
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

}

# End of Affymetrix U133 Analysis

############################################################
#
# This performs an analysis of the Affymetrix U95 spike-in
# data set
#
# As this analysis is similar to the U133 analysis, only
# differences between the analyses will be highlighted.
#
############################################################

fnames <- c("1521a99hpp_av06.CEL", "1521b99hpp_av06.CEL",
   "1521c99hpp_av06.CEL", "1521d99hpp_av06.CEL",
   "1521e99hpp_av06.CEL", "1521f99hpp_av06.CEL",
   "1521g99hpp_av06.CEL", "1521h99hpp_av06.CEL",
```

```
   "1521i99hpp_av06.CEL", "1521j99hpp_av06.CEL",
   "1521k99hpp_av06.CEL", "1521l99hpp_av06r.CEL",
   "1521m99hpp_av06.CEL", "1521n99hpp_av06.CEL",
   "1521o99hpp_av06.CEL", "1521p99hpp_av06.CEL",
   "1521q99hpp_av06.CEL", "1521r99hpp_av06.CEL",
   "1521s99hpp_av06.CEL", "1521t99hpp_av06.CEL",
   "1532a99hpp_av04.CEL", "1532b99hpp_av04.CEL",
   "1532c99hpp_av04.CEL", "1532d99hpp_av04.CEL",
   "1532e99hpp_av04.CEL", "1532f99hpp_av04.CEL",
   "1532g99hpp_av04.CEL", "1532h99hpp_av04.CEL",
   "1532i99hpp_av04.CEL", "1532j99hpp_av04.CEL",
   "1532k99hpp_av04.CEL", "1532l99hpp_av04.CEL",
   "1532m99hpp_av04.CEL", "1532n99hpp_av04.CEL",
   "1532o99hpp_av04.CEL", "1532p99hpp_av04.CEL",
   "1532q99hpp_av04.CEL", "1532r99hpp_av04.CEL",
   "1532s99hpp_av04.CEL", "1532t99hpp_av04r.CEL",
   "2353a99hpp_av08.CEL", "2353b99hpp_av08r.CEL",
   "2353d99hpp_av08.CEL", "2353e99hpp_av08.CEL",
   "2353f99hpp_av08.CEL", "2353g99hpp_av08.CEL",
   "2353h99hpp_av08.CEL", "2353i99hpp_av08.CEL",
   "2353j99hpp_av08.CEL", "2353k99hpp_av08.CEL",
   "2353l99hpp_av08.CEL", "2353m99hpp_av08.CEL",
   "2353n99hpp_av08.CEL", "2353o99hpp_av08.CEL",
   "2353p99hpp_av08.CEL", "2353q99hpp_av08.CEL",
   "2353r99hpp_av08.CEL", "2353s99hpp_av08.CEL",
   "2353t99hpp_av08.CEL")

spike.names <- c("37777_at", "684_at", "1597_at", "38734_at",
   "39058_at", "36311_at", "36889_at", "1024_at", "36202_at",
   "36085_at", "40322_at", "407_at", "1091_at", "1708_at")

spike.conc <- matrix(data=
   c(0,0.25,0.5,1,2,4,8,16,32,64,128,0,512,1024,
   0.25,0.5,1,2,4,8,16,32,64,128,256,0.25,1024,0,
   0.5,1,2,4,8,16,32,64,128,256,512,0.5,0,0.25,
   1,2,4,8,16,32,64,128,256,512,1024,1,0.25,0.5,
   2,4,8,16,32,64,128,256,512,1024,0,2,0.5,1,
   4,8,16,32,64,128,256,512,1024,0,0.25,4,1,2,
   8,16,32,64,128,256,512,1024,0,0.25,0.5,8,2,4,
   16,32,64,128,256,512,1024,0,0.25,0.5,1,16,4,8,
   32,64,128,256,512,1024,0,0.25,0.5,1,2,32,8,16,
   64,128,256,512,1024,0,0.25,0.5,1,2,4,64,16,32,
   128,256,512,1024,0,0.25,0.5,1,2,4,8,128,32,64,
   256,512,1024,0,0.25,0.5,1,2,4,8,16,256,64,128,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
```

```
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512), ncol=14,
   byrow=TRUE)

spike.group <- 1:14
names(spike.group) <- spike.names

small.fnames <- fnames

# These are the categories (experiment numbers) for the
# comparisons.  Note that one chip in Experiment 3 of the
# U95 dataset did not hybridize properly, so that there are
# only 2 chips in this comparison rather than 3.  Though
# not originally intended, this allows the assessment of the
# algorithms when differing numbers of chips are compared.
category <- c(1:20,1:20,(1:20)[-3])
cat.names <- unique(category[category!=1])

cel <- ReadAffy(filenames=small.fnames)
eset <- rma(cel)
rma.exprs <- exprs(eset)

num.large <- 14
num.small <- 14

for (i in 1:length(cat.names)) {

    data <- cbind(rma.exprs[,category==1],rma.exprs[, category==
        cat.names[i]])

    compare <- c(rep(0,sum(as.integer(category==1))),rep(1,
       sum(as.integer(category==cat.names[i]))))
    t.df<-apply(data,1,df,compare=compare)
    ttest.rma <- mt.teststat(data,compare)
    rawp0.rma <- 2*(1-pt(abs(ttest.rma),t.df))
    abs.score <- abs(rawp0.rma)
    gn <- geneNames(cel)
    index <- order(abs.score,decreasing=FALSE)

    ranking <- rank(abs.score,ties.method="min")
```

```
    ranking <- max(ranking) - ranking + 1
    ranking <- rep(1:length(unique(ranking)),times=
       as.vector(table(ranking)))
    names(ranking) <- geneNames(cel)[index]

    results <- data.frame(name=geneNames(cel)[index],
       iteration=rep(i,length(index)),rank=ranking,score=
       abs.score[index])
    outfile <- paste("RMAFoldOverallU95.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

    small.count <- sum(!is.na(match(names(ranking)[1:
       num.small],spike.names)))
    results <- data.frame(iteration=i,small.count)
    outfile <- paste("RMAFoldCountU95.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    results <- data.frame(name=spike.names,iteration=rep(i,
       length(spike.names)), expectedrank=
       fold.rank[spike.group[spike.names]], actualrank=
       ranking[spike.names])
    outfile <- paste("RMAFoldRankU95.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

}

# End of Affymetrix U95 analysis

############################################################
#
# This performs an analysis on the GeneLogic Dilution data
# set
#
# As this analysis is similar to the U133 and U95 analyses,
# only differences between the analyses will be highlighted.
#
############################################################
```

```
fnames <- c("92453hgu95a11.cel", "92454hgu95a11.cel",
   "92455hgu95a11.cel", "92456hgu95a11.cel",
   "92457hgu95a11.cel", "92458hgu95a11.cel",
   "92459hgu95a11.cel", "92460hgu95a11.cel",
   "92461hgu95a11.cel", "92462hgu95a11.cel",
   "92463hgu95a11.cel", "92464hgu95a11.cel",
   "92465hgu95a11.cel", "92466hgu95a11.cel",
   "92491hgu95a11.cel", "92492hgu95a11.cel",
   "92493hgu95a11.cel", "92494hgu95a11.cel",
   "92495hgu95a11.cel", "92496hgu95a11.cel",
   "92497hgu95a11.cel", "92498hgu95a11.cel",
   "92499hgu95a11.cel", "92500hgu95a11.cel",
   "92501hgu95a11.cel", "92503hgu95a11.cel")

fnames <- fnames[c(14,15,16,17,18,19,20,1,21,2,3,22,4,
   5,23,6,7,24,8,9,25,10,11,12,13,26)]

spike.conc <- matrix(data= c(0,0,0,0,0,0,0,0,0,0,
   0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,
   0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,
   1,1,1,1,1,1,1,1,1,1, 1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,
    2,2,2,2,2,2,2,2,2,2, 3,3,3,3,3,3,3,3,3,3,
   5,5,5,5,5,5,5,5,5,5,
   12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,
   25,25,25,25,25,25,25,25,25,25, 50,50,50,50,50,50,50,50,50,50,
    75,75,75,75,75,75,75,75,75,75,
   100,100,100,100,100,100,100,100,100,100,
   150,150,150,150,150,150,150,150,150,150), nrow=14,
   byrow=TRUE)

spike.names <- c("AFFX-BioB-5_at", "AFFX-BioB-M_at",
   "AFFX-BioB-3_at", "AFFX-BioC-5_at", "AFFX-BioC-3_at",
   "AFFX-BioDn-3_at", "AFFX-DapX-5_at", "AFFX-DapX-M_at",
   "AFFX-DapX-3_at", "AFFX-CreX-5_at")

spike.group <- 1:11
names(spike.group) <- spike.names

expanded.spike <- spike.names

expanded.group <- spike.group
names(expanded.group) <- expanded.spike

small.fnames <- fnames
```

```
# These are the categories (experiment numbers) for the
# comparisons.  Note that only experiments 9 through 12 and
# experiment 14 have a sufficient number of chips for
# comparisons using all algorithms.  Thus, the baseline
# condition will be experiment 9.  All experiments in the
# list will be compared to the baseline in turn.
category <- c(1,2,3,4,5,6,7,8,8,rep(9:12,each=3),13,13,14,14,14)
cat.names <- unique(category[category > 9])

cel <- ReadAffy(filenames=small.fnames)
eset <- rma(cel)
rma.exprs <- exprs(eset)

num.large <- 10
num.small <- 10

for (i in 1:length(cat.names)) {

    data <- cbind(rma.exprs[,category==9],rma.exprs[,
       category==cat.names[i]])

    compare <- c(rep(0,sum(as.integer(category==9))),rep(1,
       sum(as.integer(category==cat.names[i]))))
    t.df<-apply(data,1,df,compare=compare)
    ttest.rma <- mt.teststat(data,compare)
    rawp0.rma <- 2*(1-pt(abs(ttest.rma),t.df))
    abs.score <- abs(rawp0.rma)
    gn <- geneNames(cel)
    index <- order(abs.score,decreasing=TRUE)

    ranking <- rank(abs.score,ties.method="min")
    ranking <- max(ranking) - ranking + 1
    ranking <- rep(1:length(unique(ranking)),times=
       as.vector(table(ranking)))
    names(ranking) <- gn[index]

    results <- data.frame(name=gn[index],iteration=rep(i,
       length(index)),rank=ranking,score=abs.score[index])
    outfile <- paste("RMAFoldOverallGDilution.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

    posindex <- (abs.score <= 0.001)
    small.count <- sum(!is.na(match(gn[posindex], spike.names)))
```

```
    large.count <- sum(!is.na(match(gn[posindex],
       expanded.spike)))
    truepos <- large.count
    falsepos <- sum(posindex) - large.count
    falseneg <- num.large - large.count
    trueneg <- (length(abs.score) - sum(posindex)) -  falseneg
    results <- data.frame(iteration=i,truepos,falsepos,trueneg,
       falseneg)
    outfile <- paste("RMAFoldCountGDilution.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    outfile <- paste("RMAFoldRankGDilution.csv",sep="")
    results <- data.frame(name=expanded.spike,iteration=
       rep(i,length(expanded.spike)),expectedrank=
       fold.rank[abs(expanded.group[expanded.spike])],
       actualrank=ranking[expanded.spike])
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

}

# End of GeneLogic Dilution Analysis
```

A.7 RVM Analysis of Spike-In Datasets

```
#############################################################
#
# Program Name: RMAAnalysis.R
# Author: Richard Kennedy
# Date: 12/21/2007
#
# Purpose:  This program performs an automated analysis on
# several sets of data using the RMA algorithm as the
# expression summary measure.
#
# Description: This program analyzes three separate
# datasets, the Affymetrix U95 and U133 Latin Square and the
# GeneLogic Dilution data.  For each dataset, the
# appropriate data files are read and the RMA expression
# summary computed.  The RMA values are then compared using
# multiple t-tests to give measures of significance for
# differential gene expression.  One data file is created
# showing the p-values of the t-tests for all of the
# probesets on the chip, in increasing order (or decreasing
# order of significance); one data file gives the number of
# spike-in probes (from both the original Affymetrix list
# and the expanded list of McGee et al.) that are highly
# ranked; and one data file shows the actual rank based on
# the p-values versus the expected rank based on the
# concentration fold-change from the spike-in data.
# Although similar, separate computation routines are used
# for the Affymetrix U133 Latin Square, Affymetrix U95 Latin
# Square, and GeneLogic Dilution datasets due to slight
# differences in the analyses and for better readability.
#
#############################################################

# Load the affy library.  This is a standard library
# available through Bioconductor, which implements the
# functions for reading CEL files
library(affy)

# Load the nlme library.  This is a standard library
# available through CRAN, which implements the glsfit
# function for generalized least squares
library(nlme)
```

```
# This function implements the matrix square root.
# Input:  x - a p x p dimensional matrix for which the
#         square root is desired.
# Output: A p x p dimensional matrix representing
#         the matrix square root of the input.
msqrt <- function(x) {

# Find the eigen decomposition of the matrix
    eig <- eigen(x)
    eigvec <- eig$vectors

# The square root of the matrix is found by creating
# a diagonal matrix of the square roots of the eigenvalues,
# then pre- and post- multiplying this by the eigenvectors
    result <- eigvec %*% diag(sqrt(eig$values)) %*% t(eigvec)

    return(result)
}

# This function implements the logaritm of the multivariate
# gamma function.  The logarithm of the multivariate gamma
# function, rather than the function value itself, is
# returned due to the magnitude of the numbers involved.
# Formulae for the multivariate gamma function are given in
# a number of sources, e.g., Muirhead pp. 61-62.
# Input:  p - rank of the matrices over whose set the
#             multivariate gamma integral is being evaluated
#         a - degrees of freedom
# Output: A scalar containing the value of the multivariate
#             gamma function for the specified matrix rank
mvlgamma <- function(p,a) {
    result <- (p*(p-1)/4)*log(pi) + sum(lgamma(a-((1:p)-1)/2))
    return(result)
}

# This function implements the logarithm of the singular
# generalized multivariate beta type II density for a
# specified observation point. If the density cannot be
# computed, an infinite value is returned so that this point
# will be avoided in the minimization process.  The
# logarithm is used again due to the magnitude of the
# numbers involved.  The distribution function used is given
# in a variety of sources, e.g., Srivastava (2003), p. 1553.
# Input:  x - a p x p matrix representing the observed value
#             from the multivariate beta distribution
```

```
#         pval - size of the observed value matrix, which is
#             assumed to be singular
#         n1, n2 - degrees of freedom (for the "numerator"
#             and "denominator" matrices respectively, if
#             considering the multivariate beta as a product
#             of two Wishart distributions)
#         omega - the scale parameter matrix
# Output:  A scalar containing the value of the generalized
#             multivariate beta type II density at the
#             observed value x
ldmvbeta <- function(x,pval,n1,n2,omega) {

# Obtain the spectral decomposition of x
    x.eigen <- eigen(x,only.values=TRUE)

# Since the determinant of x does not exist, the singular
# multivariate beta type II distribution uses the product
# of the first n1 eigenvalues, assuming rank(x) = n1
    determ <- prod(x.eigen$values[1:min(n1,pval)])

# If the parameter values are not valid for calculating the
# logarithms in the density - which occurs if the product of
# the first n1 eigenvalues is negative or the determinant of
# I + Omega * x is negative - then return infinity, the
# largest possible value.  This will keep the point from
# being used in the minimization process
    if ((determ < 0) | (det(diag(pval) + omega %*% x) < 0)) {
        result <- Inf
    } else {

# Otherwise, return the value of the density at the
# specified point
        result <- ((n1*n1 - n1*pval)/2) * log(pi) +
           mvlgamma(pval,(n1+n2)/2) - mvlgamma(n1,n1/2) -
           mvlgamma(pval,n2/2) + (n1/2) * log(det(omega)) +
           ((n1-pval-1)/2)*log(determ) - ((n1+n2)/2) *
           log(det(diag(pval) + omega %*% x))
    }
    return(result)
}

# This function implements the logarithm of the multivariate
# beta likelihood, which computes the likelihood of a series
# of observations, with each observation having a common
# multivariate beta distribution.
```

```
# Input:  x - a vector of the parameters to be optimized.
#             The first element is the degrees of freedom
#             for the inverse Wishart prior and the second
#             and third elements are used for constructing
#             the matrix parameter for the prior.  The
#             second element is used for the diagonal
#             element of the compound symmetric matrix
#             parameter and the third is used for the off-
#             diagonal elements.
#         pval - the dimension p of a single p x p
#             observation matrix
#         mdf - the degrees of freedom for the comparison
#             being conducted, i.e. n-k
#         msigmahat - a matrix of observations for which the
#             likelihood is desired.  Each row is a single
#             observation matrix, which has been vectorized
#             from a p x p matrix to a p^2 x 1 vector.  Thus
#             the number of rows in x also equals the number
#             of observations.
#         const - a vector of constants passed to the
#             function.  This is not used in the current
#             implementation, but may be used in future
#             versions to fix certain parameters to be
#             constants rather than optimized.
# Output:
mvbetalik <- function(x,pval,mdf,msigmahat,const) {

# first make a copy of the scalar parameter of the prior,
# since R uses a form of passing by reference
    esta <- x[1]

# Now create the compound symmetric matrix for the matrix
# parameter of the prior.  Copy the second element of the
# parameter vector to the diagonal elements of the matrix,
# and the third element to the off-diagonal elements.
    estb <- diag(x[2],pval,pval)
    estb[upper.tri(estb)] <- x[3]
    estb[lower.tri(estb)] <- x[3]

# Check whether the parameters are valid for the
# multivariate beta density, with the prior being
# full rank.  The following three conditions must be met:
# (1) the degrees of freedom esta for the prior must be
#     greater than or equal to the dimension of the matrix
#     parameter for the prior, which is equal to pval.
```

```
# (2) The determinant of the matrix parameter must be
#     nonzero, signifying full rank.
# (3) The eigenvalues of the matrix parameter must be
#     positive, signifying positive definiteness.
# If the parameters are not valid, the likelihood is set
# to infinity so that these parameters will not be
# considered minimized.
    if ((esta < pval) | (det(estb) == 0) |
       any(eigen(estb,only.values=TRUE)$values <= 0)) {
        result <- Inf
    } else {

# The parameters are valid.  First, find the matrix square
# root of the matrix parameter for use in later
# calculations.
        estbhalf <- msqrt(estb)

# Iterate over all observations to find the likelihood with
# the current values of the parameters being optimized.
# Since the log likelihood is used, the total likelihood is
# the sum of the likelihoods for the individual obserations.
        estlik <- rep(0,nrow(msigmahat))
        for (i in 1:nrow(msigmahat)) {

# Get the value of the current observation and compute
# the likelihood of this one observation using the current
# value of the parameters
            onesigma <- matrix(data=msigmahat[i,],ncol=pval,
               nrow=pval)
            estlik[i] <- ldmvbeta(x=(esta+pval-1)*estbhalf %*%
               onesigma %*% t(estbhalf),m=pval,n1=mdf,n2=
               esta+pval-1, omega=diag(mdf/(esta+pval-1),pval,
               pval))

# Note that it is (a+p-1) * sqrt(B) * Sn * sqrt(B) which
# follows the multivariate F distribution, while the data
# msigmahat are for Sn only.  The likelihood for Sn can be
# obtained by multiplying by the Jacobian
# det((a+p-1) * B) ^ ((p+1)/2)
# or by adding ((p+1)/2) * log(det(a+p-1) * B) to the log
# likelihood.  This is _not_ described in the Wright and
# Simon article but is contained in their code for the
# univariate RVM method.
            estlik[i] <- estlik[i] + (pval+1)/2 *
               log(det((esta+pval-1)* estb))
```

```
        }

# If any of the individual likelihoods is infinite (i.e. not
# valid), then return a result of infinity.  Otherwise the
# total likelihood is the sum of the individual likelihoods
        if (all(is.finite(estlik))) {
            result <- sum(-estlik)
        } else {
            result <- Inf
        }
    }
    return(result)
}

# End of declared functions

#############################################################
#
# This performs an analysis of the Affymetrix U133 spike-in
# data set
#
#############################################################

# These are the filenames, which are stored in order of the
# ASCII collating sequence, as in the directory listing.
fnames <- c("12_13_02_U133A_Mer_Latin_Square_Expt10_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt10_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt10_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt11_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt11_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt11_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt12_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt12_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt12_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt13_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt13_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt13_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt14_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt14_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt14_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt1_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt1_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt1_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt2_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt2_R2.CEL",
```

```
"12_13_02_U133A_Mer_Latin_Square_Expt2_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt3_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt3_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt3_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt4_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt4_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt4_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt5_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt5_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt5_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt6_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt6_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt6_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt7_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt7_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt7_R3.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt8_R1.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt8_R2.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt8_R3.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt9_R1.CEL",
"12_13_02_U133A_Mer_Latin_Square_Expt9_R2.CEL",
   "12_13_02_U133A_Mer_Latin_Square_Expt9_R3.CEL")

# Put the filenames in numerical order
fnames <- fnames[c(16:42,1:15)]

# These are the names of the spiked-in clones for the
# original 42 probes reported by Affymetrix.  These are in
# the order given in the Affymetrix descriptor file included
# with the dataseta.  Note that there are 3 clones in each
# group of clones spiked in at the same concentration for a
# given experiment (see the Affymetrix descriptor file for
# additional information).
spike.names <- c("203508_at", "204563_at", "204513_s_at",
   "204205_at", "204959_at", "207655_s_at", "204836_at",
   "205291_at", "209795_at", "207777_s_at", "204912_at",
   "205569_at", "207160_at", "205692_s_at", "212827_at",
   "209606_at", "205267_at", "204417_at", "205398_s_at",
   "209734_at", "209354_at", "206060_s_at", "205790_at",
   "200665_s_at", "207641_at", "207540_s_at", "204430_s_at",
   "203471_s_at", "204951_at", "207968_s_at", "AFFX-r2-TagA_at",
    "AFFX-r2-TagB_at", "AFFX-r2-TagC_at", "AFFX-r2-TagD_at",
   "AFFX-r2-TagE_at", "AFFX-r2-TagF_at", "AFFX-r2-TagG_at",
   "AFFX-r2-TagH_at", "AFFX-DapX-3_at", "AFFX-LysX-3_at",
   "AFFX-PheX-3_at", "AFFX-ThrX-3_at")
```

```
# These are the concentration data for the clones in each
# experiment.  These are ordered across columns by clone
# group and across rows by experiment (or chip group).
spike.conc <- matrix(data=
   c(0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,
   0.125,0.25,0.5,1,2,4,8,16,32,64,128,256,512,0,
   0.25,0.5,1,2,4,8,16,32,64,128,256,512,0,0.125,
   0.5,1,2,4,8,16,32,64,128,256,512,0,0.125,0.25,
   1,2,4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,
   2,4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,
   4,8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,
   8,16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,
   16,32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,
   32,64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,
   64,128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,
   128,256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,
   256,512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,
   512,0,0.125,0.25,0.5,1,2,4,8,16,32,64,128,256), nrow=14,
   byrow=TRUE)

# These are the group numbers for each of the spiked-in
# clones given in the spike.names variable
spike.group <- rep(1:14,each=3)
names(spike.group) <- spike.names

# These are the names of the spiked-in clones for the
# expanded set of 64 probes reported by McGee et al.  These
# are in the order given in their article and the
# supplemental files.  Note that there are no longer 3
# clones in each group when using the expanded set.
expanded.spike <- c("200665_s_at", "203471_s_at", "203508_at",
   "204205_at", "204417_at", "204430_s_at", "204513_s_at",
   "204563_at", "204836_at", "204912_at", "204951_at",
   "204959_at", "205267_at", "205291_at", "205398_s_at",
   "205569_at", "205692_s_at", "205790_at", "206060_s_at",
   "207160_at", "207540_s_at", "207641_at", "207655_s_at",
   "207777_s_at", "207968_s_at", "208010_s_at", "209354_at",
   "209374_s_at", "209606_at", "209734_at", "209795_at",
   "212827_at", "AFFX-DapX-3_at", "AFFX-DapX-5_at",
   "AFFX-DapX-M_at", "AFFX-LysX-3_at", "AFFX-LysX-5_at",
   "AFFX-LysX-M_at", "AFFX-PheX-3_at", "AFFX-PheX-5_at",
   "AFFX-PheX-M_at", "AFFX-ThrX-3_at", "AFFX-ThrX-5_at",
   "AFFX-ThrX-M_at", "AFFX-r2-Bs-dap-3_at",
   "AFFX-r2-Bs-dap-5_at", "AFFX-r2-Bs-dap-M_at",
```

```
   "AFFX-r2-Bs-lys-3_at", "AFFX-r2-Bs-lys-5_at",
   "AFFX-r2-Bs-lys-M_at", "AFFX-r2-Bs-phe-3_at",
   "AFFX-r2-Bs-phe-5_at", "AFFX-r2-Bs-phe-M_at",
   "AFFX-r2-Bs-thr-3_s_at", "AFFX-r2-Bs-thr-5_s_at",
   "AFFX-r2-Bs-thr-M_s_at", "AFFX-r2-TagA_at",
   "AFFX-r2-TagB_at", "AFFX-r2-TagC_at", "AFFX-r2-TagD_at",
   "AFFX-r2-TagE_at", "AFFX-r2-TagF_at", "AFFX-r2-TagG_at",
   "AFFX-r2-TagH_at")

# These are the group numbers for each of the spiked-in
# clones given in the expanded.spike variable.  Positive
# numbers denote the original 42 clones reported by
# Affymetrix, negative numbers the supplemental 22 clones
# given by McGee et al., to facilitate separate analyses if
# necessary.
expanded.group <- c(8,10,1,2,6,9,1,1,3,4,10,2,6,3,7,4,5,8,8,
   5,9,9,2,4,10,-8,7,-5,6,7,3,5,13,-13,-13,14,-14,-14,14,-14,
   -14,14,-14,-14,-13,-13,-13,-14,-14,-14,-14,-14,-14,-14,-14,
   -14,11,11,11,12,12,12,13,13)
names(expanded.group) <- expanded.spike

# These are the categories (experiment numbers) to which
# each of the chips belongs
category <- rep(1:14,each=3)

# These are the categories (experiment numbers) for the
# comparisons.  By default, the baseline condition will be
# experiment 1.  All experiments in the list will be
# compared to the baseline in turn
cat.names <- unique(category[category!=1])

# This is a list of the filenames of the CEL files for this
# analysis. A separate variable is used to facilitate
# subanalyses if necessary.
small.fnames <- fnames

# These are the number of probes in the expanded and
# original list of spiked-in clones.
num.large <- 64
num.small <- 42

# loop through the list of experiments for comparison
for (i in 1:length(cat.names)) {

# This is a list of the filenames of the CEL files for this
```

```
# analysis. A separate variable is used to facilitate
# subanalyses if necessary.
    index <- category==1 | category==cat.names[i]
    small.fnames <- fnames[index]

# get the expression summary data for the comparison
    data <- ReadAffy(filenames=small.fnames)

# construct the comparison vector, used in building the
# design matrix for the glsfit function
    compare <- c(rep(0,sum(as.integer(category==1))),rep(1,
       sum(as.integer(category==cat.names[i]))))

# Initialize vectors that will be used for storing the RVM
# chi-square and p values
    abs.score <- NULL
    all.lambda <- NULL
    all.chival <- NULL

# For the Affymetrix U133 chip, the vast majority of the
# probesets contain 11 probe pairs each.  Obtain this list
# of probesets and use it for the estimation of the
# parameters of the prior.
    p <- 11
    pmidx <- pmindex(data)
    pmidx.vec <- lapply(pmidx,length)
    idx <- (pmidx.vec==p)
    pmidx <- pmidx[idx]

# The list of probesets is a vector of lists, with each list
# containing the probe IDs within the probeset.  Convert
# this into a vector of probe IDs for construction of the
# linear model and model fitting.
    upmidx <- unlist(pmidx)
    pmidx.vec <- lapply(pmidx,length)
    num <- rep(1:length(pmidx),pmidx.vec)

# Initialize matrices for storing the residual sums of
# squares from the full (ss-hat) and reduced (ss-hat-hat)
# models
    ssfull <- matrix(data=0,nrow=length(pmidx), ncol= p*p)
    ssreduced <- matrix(data=0,nrow=length(pmidx), ncol= p*p)

# Get the intensity data and log2 transform it
    intens <- log2(intensity(data))
```

```
# Loop through each probeset in the list of size 11
# probesets
    for (j in 1:length(pmidx)) {

# Get the list of probe IDs for the current probeset
        oneset <- pmidx[j]
        uoneset <- unlist(oneset)

# Get the corresponding intensities and vectorize them
# for model fitting
        oneintens <- intens[uoneset,]
        y <- as.vector(oneintens)

# Construct the design matrix with designations for
# treatment group (0 for baseline, 1 for experimental,
# derived from the compare vector), chip (numbered 1
# through the number of chips), probeset (which is set to 1 #
   since the fitting is done separately for each probeset),
# and probe (numbered 1 through 11).
        treat <- as.factor(rep(compare,each= nrow(oneintens)))
        chip <- as.factor(rep(1:length(compare), each=
           nrow(oneintens)))
        probeset <- as.factor(rep(1,length(y)))
        probe <- as.factor(rep(1:p,ncol(oneintens)))

# Combine the intensity data and design matrix into a
# single data frame for model fitting
        affydata <- data.frame(y,treat,chip,probeset, probe)

# Fit the full model with treatment and probe effects
        glsfit <- gls(y ~ treat + probe, correlation =
           corCompSymm(form = ~ 1 | chip), data=affydata,
           method="ML",control=glsControl(opt="optim"))

# Get the fitted values
        yfitted <- matrix(data=fitted(glsfit),nrow=
           nrow(oneintens),ncol=ncol(oneintens))
        yfitted <- t(yfitted)

# Calculate the residual sums of squares for the current
# probeset, which is vectorized and stored as a row in the
# ssfull matrix
        ss <- t(t(oneintens) - yfitted) %*% (t(oneintens) -
           yfitted)
```

```
        ssfull[j,] <- as.vector(ss)

# Fit the reduced model with only probe effects
        glsfit <- gls(y ~ probe, correlation = corCompSymm(form
           = ~ 1 | chip), data=affydata,
           method="ML",control=glsControl(opt="optim"))

# Get the fitted values
        yfitted <- matrix(data=fitted(glsfit),nrow=
           nrow(oneintens),ncol=ncol(oneintens))
        yfitted <- t(yfitted)

# Calculate the residual sums of squares, which is
# vectorized and stored as a row in the ssreduced matrix
        ss <- t(t(oneintens) - yfitted) %*% (t(oneintens) -
           yfitted)
        ssreduced[j,] <- as.vector(ss)
    }

# Calculate the residual degrees of freedom
    resdf <- (length(compare) - length(unique(compare)))

# Estimate the error (sigma) matrix, which is the residual
# sums of squares divided by the residual degrees of freedom
# using the full model
    sigmahat <- ssfull/resdf

# Calculate the average of the estimates for sigma, which
# will be used as the starting value for optimizing the
# matrix parameter of the prior
    meansigma <- colMeans(sigmahat)

# Optimize using the optim function to obtain maximum
# likelihood estimates of the parameters
    startvals <- c(p+1,meansigma[1],meansigma[2])
    optimresult <- optim(par=startvals,fn=mvbetalik,method=
       "Nelder-Mead",pval=p,mdf=resdf,msigmahat=sigmahat,
       const=c(10,2,3),control=list(maxit=1000))

# Give warning messages if the optimization algorithm did
# not converge, but proceed with the likelihood ratio test
    if (optimresult$convergence !=0 ) {
        writeLines(sprintf("Problems with convergence in
           iteration %i",i))
    }
```

```
    if (optimresult$convergence == 1) {
        writeLines("Maximum number of iterations reached,
           consider increasing")
    }

# Get the maximum likelihood estimates for the parameters of
# the prior
    result <- optimresult$par
    aval <- result[1]
    bmat <- diag(result[2],p,p)
    bmat[upper.tri(bmat)] <- result[3]
    bmat[lower.tri(bmat)] <- result[3]

# Compute the inverse of the matrix parameter, which will be
# used to adjust the residual sums of squares in calculating
# the value of the likelihood ratio test
    bvec <- as.vector(solve(bmat))

# Initialize the vectors for showing the the ratio of
# determinants for the full and reduced models (lambda),
# the chi-square value (testval), and the p-value (score)
    lambda <- rep(0,length(pmidx))
    testval <- rep(0,length(pmidx))
    sshat <- ssfull
    sshathat <- ssreduced


    for (j in 1:length(pmidx)) {
            sshat <- matrix(data=ssfull[j,] + bvec,p,p)
            sshathat <- matrix(ssreduced[j,] + bvec,p,p)
            lambda[j] <- det(sshat) / det(sshathat)
            testval[j] <- ifelse(lambda[j]>0, -(resdf+
               aval-p-1) * log(lambda[j]),0)
    }
    score <- ifelse(testval<0,1,1-pchisq(testval,df=4))

# Write out the results for this probeset for later use
    index <- order(testval,decreasing=TRUE)
    gn <- names(pmidx)
    results <- data.frame(name=gn[index],iteration=
       rep(i,length(index)),lambda=lambda[index],testval=
       testval[index],score=score[index])
    outfile <- paste("RVM",p,"FoldOverallU133.csv", sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))
```

```
    names(score) <- gn

    abs.score <- score
    all.lambda <- lambda
    all.testval <- testval

# Repeat the calculations for the probesets with 20 probes
# and the probeset with 16 probes.  In these cases, there
# are not enough probesets to obtain an accurate estimate of #
   the prior through fitting the multivariate beta
# distribution.  However, assuming the probesets with 11
# probes are representative of the remaining probes, the
# parameter estimates from the previous step can be used in
# constructing the estimates of the prior for the probesets
# with 20 and 16 probes
    for (p in c(20,16)) {
        aval <- p
        bmat <- diag(result[2],p,p)
        bmat[upper.tri(bmat)] <- result[3]
        bmat[lower.tri(bmat)] <- result[3]
        bvec <- as.vector(solve(bmat))
        pmidx <- pmindex(data)
        pmidx.vec <- lapply(pmidx,length)
        idx <- (pmidx.vec==p)
        pmidx <- pmidx[idx]
        upmidx <- unlist(pmidx)
        pmidx.vec <- lapply(pmidx,length)
        num <- rep(1:length(pmidx),pmidx.vec)
        ssfull <- matrix(data=0,nrow=length(pmidx),ncol=p*p)
        ssreduced <- matrix(data=0,nrow=length(pmidx),ncol=p*p)

        for (j in 1:length(pmidx)) {
            oneset <- pmidx[j]
            uoneset <- unlist(oneset)
            oneintens <- intens[uoneset,]
            y <- as.vector(oneintens)
            treat <- as.factor(rep(compare,each=
               nrow(oneintens)))
            chip <- as.factor(rep(1:length(compare),each=
               nrow(oneintens)))
            probeset <- as.factor(rep(1,length(y)))
            probe <- as.factor(rep(1:p,ncol(oneintens)))
            affydata <- data.frame(y,treat,chip,probeset, probe)
            glsfit <- gls(y ~ treat + probe, correlation =
               corCompSymm(form = ~ 1 | chip), data=affydata,
```

```
      method="ML",control=glsControl(opt="optim"))
    yfitted <- matrix(data=fitted(glsfit),nrow=
      nrow(oneintens),ncol=ncol(oneintens))
    yfitted <- t(yfitted)
    ss <- t(t(oneintens) - yfitted) %*% (t(oneintens) -
      yfitted)
    ssfull[j,] <- as.vector(ss)
    glsfit <- gls(y ~ probe, correlation =
      corCompSymm(form = ~ 1 | chip), data=affydata,
      method="ML",control=glsControl(opt="optim"))
    yfitted <- matrix(data=fitted(glsfit),nrow=
      nrow(oneintens),ncol=ncol(oneintens))
    yfitted <- t(yfitted)
    ss <- t(t(oneintens) - yfitted) %*% (t(oneintens) -
      yfitted)
    ssreduced[j,] <- as.vector(ss)
}

resdf <- (length(compare) - length(unique(compare)))
sigmahat <- ssfull/resdf
meansigma <- colMeans(sigmahat)

lambda <- rep(0,length(pmidx))
testval <- rep(0,length(pmidx))
sshat <- ssfull
sshathat <- ssreduced


for (j in 1:length(pmidx)) {
        sshat <- matrix(data=ssfull[j,] + bvec,p,p)
        sshathat <- matrix(ssreduced[j,] + bvec,p,p)
        lambda[j] <- det(sshat) / det(sshathat)
        testval[j] <- ifelse(lambda[j]>0,-(resdf+
          aval-p-1) * log(lambda[j]),0)
}

    score <- ifelse(testval<0,1,1-pchisq(testval, df=4))

index <- order(testval,decreasing=TRUE)
gn <- names(pmidx)
results <- data.frame(name=gn[index],iteration=rep(i,
  length(index)),lambda=lambda[index],testval=
  testval[index],score=score[index])
outfile <- paste("RVM",p,"FoldOverallU133.csv", sep="")
```

```
        write.table(results,file=outfile,sep=",",
           row.names=FALSE,col.names=(i==1),append=(i!=1))
        names(score) <- gn
        abs.score <- c(abs.score,score)
        all.lambda <- c(all.lambda,lambda)
        all.testval <- c(all.testval,testval)
    }

# rank the scores in decreasing order, with ties being
# assigned rank equal to the smallest rank of the group of
# ties
    index <- order(abs.score,decreasing=FALSE)
    ranking <- rank(abs.score,ties.method="min")[index]
    gn <- names(abs.score)

# create a data frame with the probeset (gene) names, rank,
# and score in order of increasing p-values
    results <- data.frame(name=gn[index],iteration=rep(i,
       length(index)),rank=ranking,testval=all.testval[index],
       score=abs.score[index])
    outfile <- paste("RVMFoldOverallU133.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

# count the number of spike-in probes ranked in the top 42
# (for the original list) or top 64 (for the expanded list)
# of probesets
    small.count <- sum(!is.na(match(names(ranking)[1:
       num.small],spike.names)))
    large.count <- sum(!is.na(match(names(ranking)[1:
       num.large],expanded.spike)))
    results <- data.frame(iteration=i,small.count, large.count)
    outfile <- paste("RVMFoldCountU133.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

# create a data frame with the expected rank (based on fold
# change of the spike-in concentration) of the expanded list
# of spike-in probesets to compare to the actual rank (based
# on the S-Score values)
    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
       as.vector(table(fold.rank)))
    outfile <- paste("RVMFoldRankU133.csv",sep="")
```

```
    results <- data.frame(name=expanded.spike,iteration=
       rep(i,length(expanded.spike)),expectedrank=
       fold.rank[abs(expanded.group[expanded.spike])],
       actualrank=ranking[expanded.spike])
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

}

# End of Affymetrix U133 Analysis

############################################################
#
# This performs an analysis of the Affymetrix U95 spike-in
# data set
#
# As this analysis is similar to the U133 analysis, only
# differences between the analyses will be highlighted.
#
############################################################

fnames <- c("1521a99hpp_av06.CEL", "1521b99hpp_av06.CEL",
   "1521c99hpp_av06.CEL", "1521d99hpp_av06.CEL",
   "1521e99hpp_av06.CEL", "1521f99hpp_av06.CEL",
   "1521g99hpp_av06.CEL", "1521h99hpp_av06.CEL",
   "1521i99hpp_av06.CEL", "1521j99hpp_av06.CEL",
   "1521k99hpp_av06.CEL", "1521l99hpp_av06r.CEL",
   "1521m99hpp_av06.CEL", "1521n99hpp_av06.CEL",
   "1521o99hpp_av06.CEL", "1521p99hpp_av06.CEL",
   "1521q99hpp_av06.CEL", "1521r99hpp_av06.CEL",
   "1521s99hpp_av06.CEL", "1521t99hpp_av06.CEL",
   "1532a99hpp_av04.CEL", "1532b99hpp_av04.CEL",
   "1532c99hpp_av04.CEL", "1532d99hpp_av04.CEL",
   "1532e99hpp_av04.CEL", "1532f99hpp_av04.CEL",
   "1532g99hpp_av04.CEL", "1532h99hpp_av04.CEL",
   "1532i99hpp_av04.CEL", "1532j99hpp_av04.CEL",
   "1532k99hpp_av04.CEL", "1532l99hpp_av04.CEL",
   "1532m99hpp_av04.CEL", "1532n99hpp_av04.CEL",
   "1532o99hpp_av04.CEL", "1532p99hpp_av04.CEL",
   "1532q99hpp_av04.CEL", "1532r99hpp_av04.CEL",
   "1532s99hpp_av04.CEL", "1532t99hpp_av04r.CEL",
   "2353a99hpp_av08.CEL", "2353b99hpp_av08r.CEL",
   "2353d99hpp_av08.CEL", "2353e99hpp_av08.CEL",
   "2353f99hpp_av08.CEL", "2353g99hpp_av08.CEL",
   "2353h99hpp_av08.CEL", "2353i99hpp_av08.CEL",
```

```
   "2353j99hpp_av08.CEL", "2353k99hpp_av08.CEL",
   "2353l99hpp_av08.CEL", "2353m99hpp_av08.CEL",
   "2353n99hpp_av08.CEL", "2353o99hpp_av08.CEL",
   "2353p99hpp_av08.CEL", "2353q99hpp_av08.CEL",
   "2353r99hpp_av08.CEL", "2353s99hpp_av08.CEL",
   "2353t99hpp_av08.CEL")

spike.names <- c("37777_at", "684_at", "1597_at", "38734_at",
   "39058_at", "36311_at", "36889_at", "1024_at", "36202_at",
   "36085_at", "40322_at", "407_at", "1091_at", "1708_at")

spike.conc <- matrix(data=
   c(0,0.25,0.5,1,2,4,8,16,32,64,128,0,512,1024,
   0.25,0.5,1,2,4,8,16,32,64,128,256,0.25,1024,0,
   0.5,1,2,4,8,16,32,64,128,256,512,0.5,0,0.25,
   1,2,4,8,16,32,64,128,256,512,1024,1,0.25,0.5,
   2,4,8,16,32,64,128,256,512,1024,0,2,0.5,1,
   4,8,16,32,64,128,256,512,1024,0,0.25,4,1,2,
   8,16,32,64,128,256,512,1024,0,0.25,0.5,8,2,4,
   16,32,64,128,256,512,1024,0,0.25,0.5,1,16,4,8,
   32,64,128,256,512,1024,0,0.25,0.5,1,2,32,8,16,
   64,128,256,512,1024,0,0.25,0.5,1,2,4,64,16,32,
   128,256,512,1024,0,0.25,0.5,1,2,4,8,128,32,64,
   256,512,1024,0,0.25,0.5,1,2,4,8,16,256,64,128,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   512,1024,0,0.25,0.5,1,2,4,8,16,32,512,128,256,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512,
   1024,0,0.25,0.5,1,2,4,8,16,32,64,1024,256,512), ncol=14,
   byrow=TRUE)

spike.group <- 1:14
names(spike.group) <- spike.names

# These are the categories (experiment numbers) for the
# comparisons.  Note that one chip in Experiment 3 of the
# U95 dataset did not hybridize properly, so that there are
# only 2 chips in this comparison rather than 3.  Though
# not originally intended, this allows the assessment of the
# algorithms when differing numbers of chips are compared.
category <- c(1:20,1:20,(1:20)[-3])
cat.names <- unique(category[category!=1])
```

```
num.large <- 14
num.small <- 14

for (i in 1:length(cat.names)) {
    index <- category==1 | category==cat.names[i]
    small.fnames <- fnames[index]
    data <- ReadAffy(filenames=small.fnames)
    compare <- c(rep(0,sum(as.integer(category==1))),rep(1,
       sum(as.integer(category==cat.names[i]))))

    abs.score <- NULL
    all.lambda <- NULL
    all.chival <- NULL

# For the Affymetrix U95 chip, the vast majority of
# probesets have 16 probes.  Use the list of probesets
# having 16 probes for fitting the multivariate beta
# distribution and obtaining maximum likelihood estimates
# of the parameters for the prior
    p <- 16
    pmidx <- pmindex(data)
    pmidx.vec <- lapply(pmidx,length)
    idx <- (pmidx.vec==p)
    pmidx <- pmidx[idx]
    upmidx <- unlist(pmidx)
    pmidx.vec <- lapply(pmidx,length)
    num <- rep(1:length(pmidx),pmidx.vec)
    ssfull <- matrix(data=0,nrow=length(pmidx),ncol=p*p)
    ssreduced <- matrix(data=0,nrow=length(pmidx),ncol= p*p)
    intens <- log2(intensity(data))

    for (j in 1:length(pmidx)) {
        oneset <- pmidx[j]
        uoneset <- unlist(oneset)
        oneintens <- intens[uoneset,]
        y <- as.vector(oneintens)
        treat <- as.factor(rep(compare,each= nrow(oneintens)))
        chip <- as.factor(rep(1:length(compare),each=
           nrow(oneintens)))
        probeset <- as.factor(rep(1,length(y)))
        probe <- as.factor(rep(1:p,ncol(oneintens)))
        affydata <- data.frame(y,treat,chip,probeset, probe)
        glsfit <- gls(y ~ treat + probe, correlation =
           corCompSymm(form = ~ 1 | chip), data=affydata,
```

```
      method="ML",control=glsControl(opt="optim"))
    yfitted <- matrix(data=fitted(glsfit),nrow=
      nrow(oneintens),ncol=ncol(oneintens))
    yfitted <- t(yfitted)
    ss <- t(t(oneintens) - yfitted) %*% (t(oneintens) -
      yfitted)
    ssfull[j,] <- as.vector(ss)
    glsfit <- gls(y ~ probe, correlation = corCompSymm(form
      = ~ 1 | chip), data=affydata,
      method="ML",control=glsControl(opt="optim"))
    yfitted <- matrix(data=fitted(glsfit),nrow=
      nrow(oneintens),ncol=ncol(oneintens))
    yfitted <- t(yfitted)
    ss <- t(t(oneintens) - yfitted) %*% (t(oneintens) -
      yfitted)
    ssreduced[j,] <- as.vector(ss)
}

resdf <- (length(compare) - length(unique(compare)))
sigmahat <- ssfull/resdf
meansigma <- colMeans(sigmahat)

startvals <- c(p+1,meansigma[1],meansigma[2])
optimresult <- optim(par=startvals,fn=mvbetalik,method=
   "Nelder-Mead",pval=p,mdf=resdf,msigmahat=sigmahat,
   const=c(10,2,3),control=list(maxit=1000))

if (optimresult$convergence !=0 ) {
    writeLines(sprintf("Problems with convergence in
      iteration %i",i))
}
if (optimresult$convergence == 1) {
    writeLines("Maximum number of iterations reached,
      consider increasing")
}

result <- optimresult$par
aval <- result[1]
bmat <- diag(result[2],p,p)
bmat[upper.tri(bmat)] <- result[3]
bmat[lower.tri(bmat)] <- result[3]
bvec <- as.vector(solve(bmat))
lambda <- rep(0,length(pmidx))
testval <- rep(0,length(pmidx))
sshat <- ssfull
```

```
    sshathat <- ssreduced


    for (j in 1:length(pmidx)) {
            sshat <- matrix(data=ssfull[j,] + bvec,p,p)
            sshathat <- matrix(ssreduced[j,] + bvec,p,p)
            lambda[j] <- det(sshat) / det(sshathat)
            testval[j] <- ifelse(lambda[j]>0,-(resdf+ aval-p-1)
               * log(lambda[j]),0)
    }
    score <- ifelse(testval<0,1,1-pchisq(testval,df=4))

    index <- order(testval,decreasing=TRUE)
    gn <- names(pmidx)
    results <- data.frame(name=gn[index],iteration=rep(i,
       length(index)),lambda=lambda[index],testval=
       testval[index],score=score[index])
    outfile <- paste("RVM",p,"FoldOverallU95.csv", sep="")
    write.table(results,file=outfile,sep=",",
       row.names=FALSE,col.names=(i==1),append=(i!=1))
    names(score) <- gn
    abs.score <- score
    all.lambda <- lambda
    all.testval <- testval

# For the Affymetrix U95 chip, the number of probesets with
# 13, 14, 15, and 20 probes per probeset are generally
# sufficient for model fitting of the intensities, though
# not sufficiently large for estimating the values of the
# prior.  Use the values of the prior from the previous step
# to obtain the likelihood ratio test statistics for these
# probesets.
    for (p in c(13,14,15,20)) {
        aval <- p
        bmat <- diag(result[2],p,p)
        bmat[upper.tri(bmat)] <- result[3]
        bmat[lower.tri(bmat)] <- result[3]
        bvec <- as.vector(solve(bmat))
        pmidx <- pmindex(data)
        pmidx.vec <- lapply(pmidx,length)
        idx <- (pmidx.vec==p)
        pmidx <- pmidx[idx]
        upmidx <- unlist(pmidx)
        pmidx.vec <- lapply(pmidx,length)
        num <- rep(1:length(pmidx),pmidx.vec)
```

```
ssfull <- matrix(data=0,nrow=length(pmidx),ncol=p*p)
ssreduced <- matrix(data=0,nrow=length(pmidx),ncol= p*p)

for (j in 1:length(pmidx)) {
    oneset <- pmidx[j]
    uoneset <- unlist(oneset)
    oneintens <- intens[uoneset,]
    y <- as.vector(oneintens)
    treat <- as.factor(rep(compare,each=
       nrow(oneintens)))
    chip <- as.factor(rep(1:length(compare),each=
       nrow(oneintens)))
    probeset <- as.factor(rep(1,length(y)))
    probe <- as.factor(rep(1:p,ncol(oneintens)))
    affydata <- data.frame(y,treat,chip,probeset, probe)
    glsfit <- gls(y ~ treat + probe, correlation =
       corCompSymm(form = ~ 1 | chip), data=affydata,
       method="ML",control=glsControl(opt="optim"))
    yfitted <- matrix(data=fitted(glsfit),nrow=
       nrow(oneintens),ncol=ncol(oneintens))
    yfitted <- t(yfitted)
    ss <- t(t(oneintens) - yfitted) %*% (t(oneintens) -
       yfitted)
    ssfull[j,] <- as.vector(ss)
    glsfit <- gls(y ~ probe, correlation =
       corCompSymm(form = ~ 1 | chip), data=affydata,
       method="ML",control=glsControl(opt="optim"))
    yfitted <- matrix(data=fitted(glsfit),nrow=
       nrow(oneintens),ncol=ncol(oneintens))
    yfitted <- t(yfitted)
    ss <- t(t(oneintens) - yfitted) %*% (t(oneintens) -
       yfitted)
    ssreduced[j,] <- as.vector(ss)
}

resdf <- (length(compare) - length(unique(compare)))
sigmahat <- ssfull/resdf
meansigma <- colMeans(sigmahat)

lambda <- rep(0,length(pmidx))
testval <- rep(0,length(pmidx))
sshat <- ssfull
sshathat <- ssreduced
```

```
    for (j in 1:length(pmidx)) {
            sshat <- matrix(data=ssfull[j,] + bvec,p,p)
            sshathat <- matrix(ssreduced[j,] + bvec,p,p)
            lambda[j] <- det(sshat) / det(sshathat)
            testval[j] <- ifelse(lambda[j]>0,-(resdf+
               aval-p-1) * log(lambda[j]),0)
    }

        score <- ifelse(testval<0,1,1-pchisq(testval, df=4))

    index <- order(testval,decreasing=TRUE)
    gn <- names(pmidx)
    results <- data.frame(name=gn[index],iteration=
       rep(i,length(index)),lambda=lambda[index],testval=
       testval[index],score=score[index])
    outfile <- paste("RVM",p,"FoldOverallU95.csv", sep="")
    write.table(results,file=outfile,sep=",",
       row.names=FALSE,col.names=(i==1),append=(i!=1))
    names(score) <- gn
    abs.score <- c(abs.score,score)
    all.lambda <- c(all.lambda,lambda)
    all.testval <- c(all.testval,testval)
}

index <- order(abs.score,decreasing=FALSE)
ranking <- rank(abs.score,ties.method="min")[index]
gn <- names(abs.score)
results <- data.frame(name=gn[index],iteration=rep(i,
   length(index)),rank=ranking,testval=all.testval[index],
   score=abs.score[index])
outfile <- paste("RVMFoldOverallU95.csv",sep="")
write.table(results,file=outfile,sep=",",row.names=
   FALSE,col.names=(i==1),append=(i!=1))
small.count <- sum(!is.na(match(names(ranking)[1:
   num.small],spike.names)))
results <- data.frame(iteration=i,small.count)
outfile <- paste("RVMFoldCountU95.csv",sep="")
write.table(results,file=outfile,sep=",",row.names=
   FALSE,col.names=(i==1),append=(i!=1))
fold.change <- spike.conc[i+1,] / spike.conc[1,]
fold.rank <- rank(fold.change,ties.method="min")
fold.rank <- rep(length(unique(fold.rank)):1,times=
   as.vector(table(fold.rank)))
outfile <- paste("RVMFoldRankU95.csv",sep="")
```

```
    results <- data.frame(name=spike.names,iteration=rep(i,
       length(spike.names)),expectedrank=
       fold.rank[abs(spike.group[spike.names])],actualrank=
       ranking[spike.names])
    write.table(results,file=outfile,sep=",",row.names=
       FALSE,col.names=(i==1),append=(i!=1))

}

# End of Affymetrix U95 analysis

############################################################
#
# This performs an analysis on the GeneLogic Dilution data
# set
#
# As this analysis is similar to the U133 and U95 analyses,
# only differences between the analyses will be highlighted.
#
############################################################

fnames <- c("92453hgu95a11.cel", "92454hgu95a11.cel",
   "92455hgu95a11.cel", "92456hgu95a11.cel",
   "92457hgu95a11.cel", "92458hgu95a11.cel",
   "92459hgu95a11.cel", "92460hgu95a11.cel",
   "92461hgu95a11.cel", "92462hgu95a11.cel",
   "92463hgu95a11.cel", "92464hgu95a11.cel",
   "92465hgu95a11.cel", "92466hgu95a11.cel",
   "92491hgu95a11.cel", "92492hgu95a11.cel",
   "92493hgu95a11.cel", "92494hgu95a11.cel",
   "92495hgu95a11.cel", "92496hgu95a11.cel",
   "92497hgu95a11.cel", "92498hgu95a11.cel",
   "92499hgu95a11.cel", "92500hgu95a11.cel",
   "92501hgu95a11.cel", "92503hgu95a11.cel")

fnames <- fnames[c(14,15,16,17,18,19,20,1,21,2,3,22,4,5,23,
   6,7,24,8,9,25,10,11,12,13,26)]

spike.conc <- matrix(data= c(0,0,0,0,0,0,0,0,0,0,
   0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,
   0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,0.75,
   1,1,1,1,1,1,1,1,1,1, 1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,1.5,
    2,2,2,2,2,2,2,2,2,2, 3,3,3,3,3,3,3,3,3,3,
   5,5,5,5,5,5,5,5,5,5,
   12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,12.5,
```

```
   25,25,25,25,25,25,25,25,25,25, 50,50,50,50,50,50,50,50,50,50,
    75,75,75,75,75,75,75,75,75,75,
   100,100,100,100,100,100,100,100,100,100,
   150,150,150,150,150,150,150,150,150,150), nrow=14,
   byrow=TRUE)

spike.names <- c("AFFX-BioB-5_at", "AFFX-BioB-M_at",
   "AFFX-BioB-3_at", "AFFX-BioC-5_at", "AFFX-BioC-3_at",
   "AFFX-BioDn-3_at", "AFFX-DapX-5_at", "AFFX-DapX-M_at",
   "AFFX-DapX-3_at", "AFFX-CreX-5_at")

spike.group <- 1:11
names(spike.group) <- spike.names

expanded.spike <- spike.names

expanded.group <- spike.group
names(expanded.group) <- expanded.spike

# These are the categories (experiment numbers) for the
# comparisons.  Note that only experiments 9 through 12 and
# experiment 14 have a sufficient number of chips for
# comparisons using all algorithms.  Thus, the baseline
# condition will be experiment 9.  All experiments in the
# list will be compared to the baseline in turn.
category <- c(1,2,3,4,5,6,7,8,8,rep(9:12,each=3),13,13,14,14,14)
cat.names <- unique(category[category > 9])

num.large <- 10
num.small <- 10

for (i in 1:length(cat.names)) {
    index <- category==1 | category==cat.names[i]
    small.fnames <- fnames[index]
    data <- ReadAffy(filenames=small.fnames)
    compare <- c(rep(0,sum(as.integer(category==1))),rep(1,
       sum(as.integer(category==cat.names[i]))))

    abs.score <- NULL
    all.lambda <- NULL
    all.chival <- NULL
    p <- 16
    pmidx <- pmindex(data)
    pmidx.vec <- lapply(pmidx,length)
    idx <- (pmidx.vec==p)
```

```
pmidx <- pmidx[idx]
upmidx <- unlist(pmidx)
pmidx.vec <- lapply(pmidx,length)
num <- rep(1:length(pmidx),pmidx.vec)
ssfull <- matrix(data=0,nrow=length(pmidx),ncol=p*p)
ssreduced <- matrix(data=0,nrow=length(pmidx),ncol= p*p)
intens <- log2(intensity(data))

for (j in 1:length(pmidx)) {
    oneset <- pmidx[j]
    uoneset <- unlist(oneset)
    oneintens <- intens[uoneset,]
    y <- as.vector(oneintens)
    treat <- as.factor(rep(compare,each= nrow(oneintens)))
    chip <- as.factor(rep(1:length(compare),each=
       nrow(oneintens)))
    probeset <- as.factor(rep(1,length(y)))
    probe <- as.factor(rep(1:p,ncol(oneintens)))
    affydata <- data.frame(y,treat,chip,probeset, probe)
    glsfit <- gls(y ~ treat + probe, correlation =
       corCompSymm(form = ~ 1 | chip), data=affydata,
       method="ML",control=glsControl(opt="optim"))
    yfitted <- matrix(data=fitted(glsfit),nrow=
       nrow(oneintens),ncol=ncol(oneintens))
    yfitted <- t(yfitted)
    ss <- t(t(oneintens) - yfitted) %*% (t(oneintens) -
       yfitted)
    ssfull[j,] <- as.vector(ss)
    glsfit <- gls(y ~ probe, correlation = corCompSymm(form
       = ~ 1 | chip), data=affydata,
       method="ML",control=glsControl(opt="optim"))
    yfitted <- matrix(data=fitted(glsfit),nrow=
       nrow(oneintens),ncol=ncol(oneintens))
    yfitted <- t(yfitted)
    ss <- t(t(oneintens) - yfitted) %*% (t(oneintens) -
       yfitted)
    ssreduced[j,] <- as.vector(ss)
}

resdf <- (length(compare) - length(unique(compare)))
sigmahat <- ssfull/resdf
meansigma <- colMeans(sigmahat)

startvals <- c(p+1,meansigma[1],meansigma[2])
```

```
optimresult <- optim(par=startvals,fn=mvbetalik,
   method="Nelder-Mead",pval=p,mdf=resdf,msigmahat=
   sigmahat,const=c(10,2,3),control=list(maxit=1000))

if (optimresult$convergence !=0 ) {
    writeLines(sprintf("Problems with convergence in
       iteration %i",i))
}
if (optimresult$convergence == 1) {
    writeLines("Maximum number of iterations reached,
       consider increasing")
}

result <- optimresult$par
aval <- result[1]
bmat <- diag(result[2],p,p)
bmat[upper.tri(bmat)] <- result[3]
bmat[lower.tri(bmat)] <- result[3]
bvec <- as.vector(solve(bmat))
lambda <- rep(0,length(pmidx))
testval <- rep(0,length(pmidx))
sshat <- ssfull
sshathat <- ssreduced

for (j in 1:length(pmidx)) {
        sshat <- matrix(data=ssfull[j,] + bvec,p,p)
        sshathat <- matrix(ssreduced[j,] + bvec,p,p)
        lambda[j] <- det(sshat) / det(sshathat)
        testval[j] <- ifelse(lambda[j]>0,-(resdf+ aval-p-1)
           * log(lambda[j]),0)
}
score <- ifelse(testval<0,1,1-pchisq(testval,df=4))

index <- order(testval,decreasing=TRUE)
gn <- names(pmidx)
results <- data.frame(name=gn[index],iteration=rep(i,
   length(index)),lambda=lambda[index],testval=
   testval[index],score=score[index])
outfile <- paste("RVM",p,
   "FoldOverallGLDilution.csv",sep="")
write.table(results,file=outfile,sep=",",
   row.names=FALSE,col.names=(i==1),append=(i!=1))
names(score) <- gn
abs.score <- score
all.lambda <- lambda
```

```
all.testval <- testval

for (p in c(13,14,15,20)) {
    aval <- p
    bmat <- diag(result[2],p,p)
    bmat[upper.tri(bmat)] <- result[3]
    bmat[lower.tri(bmat)] <- result[3]
    bvec <- as.vector(solve(bmat))
    pmidx <- pmindex(data)
    pmidx.vec <- lapply(pmidx,length)
    idx <- (pmidx.vec==p)
    pmidx <- pmidx[idx]
    upmidx <- unlist(pmidx)
    pmidx.vec <- lapply(pmidx,length)
    num <- rep(1:length(pmidx),pmidx.vec)
    ssfull <- matrix(data=0,nrow=length(pmidx),ncol=p*p)
    ssreduced <- matrix(data=0,nrow=length(pmidx),ncol= p*p)

    for (j in 1:length(pmidx)) {
        oneset <- pmidx[j]
        uoneset <- unlist(oneset)
        oneintens <- intens[uoneset,]
        y <- as.vector(oneintens)
        treat <- as.factor(rep(compare,each=
           nrow(oneintens)))
        chip <- as.factor(rep(1:length(compare),each=
           nrow(oneintens)))
        probeset <- as.factor(rep(1,length(y)))
        probe <- as.factor(rep(1:p,ncol(oneintens)))
        affydata <- data.frame(y,treat,chip,probeset, probe)
        glsfit <- gls(y ~ treat + probe, correlation =
           corCompSymm(form = ~ 1 | chip), data=affydata,
           method="ML",control=glsControl(opt="optim"))
        yfitted <- matrix(data=fitted(glsfit),nrow=
           nrow(oneintens),ncol=ncol(oneintens))
        yfitted <- t(yfitted)
        ss <- t(t(oneintens) - yfitted) %*% (t(oneintens) -
           yfitted)
        ssfull[j,] <- as.vector(ss)
        glsfit <- gls(y ~ probe, correlation =
           corCompSymm(form = ~ 1 | chip), data=affydata,
           method="ML",control=glsControl(opt="optim"))
        yfitted <- matrix(data=fitted(glsfit),nrow=
           nrow(oneintens),ncol=ncol(oneintens))
        yfitted <- t(yfitted)
```

```
        ss <- t(t(oneintens) - yfitted) %*% (t(oneintens) -
           yfitted)
        ssreduced[j,] <- as.vector(ss)
    }

    resdf <- (length(compare) - length(unique(compare)))
    sigmahat <- ssfull/resdf
    meansigma <- colMeans(sigmahat)

    lambda <- rep(0,length(pmidx))
    testval <- rep(0,length(pmidx))
    sshat <- ssfull
    sshathat <- ssreduced


    for (j in 1:length(pmidx)) {
            sshat <- matrix(data=ssfull[j,] + bvec,p,p)
            sshathat <- matrix(ssreduced[j,] + bvec,p,p)
            lambda[j] <- det(sshat) / det(sshathat)
            testval[j] <- ifelse(lambda[j]>0,-(resdf+
               aval-p-1) * log(lambda[j]),0)
    }

    score <- ifelse(testval<0,1,1-pchisq(testval,df=4))

    index <- order(testval,decreasing=TRUE)
    gn <- names(pmidx)
    results <- data.frame(name=gn[index],iteration=
       rep(i,length(index)),lambda=lambda[index],testval=
       testval[index],score=score[index])
    outfile <- paste("RVM",p,
       "FoldOverallGLDilution.csv",sep="")
    write.table(results,file=outfile,sep=",",
       row.names=FALSE,col.names=(i==1),append=(i!=1))
    names(score) <- gn
    abs.score <- c(abs.score,score)
    all.lambda <- c(all.lambda,lambda)
    all.testval <- c(all.testval,testval)
}

index <- order(abs.score,decreasing=FALSE)
ranking <- rank(abs.score,ties.method="min")[index]
gn <- names(abs.score)
results <- data.frame(name=gn[index],iteration=rep(i,
   length(index)),rank=ranking,testval=all.testval[index],
```

```
      score=abs.score[index])
    outfile <- paste( "RVMFoldOverallGLDilution.csv", sep="")
    write.table(results,file=outfile,sep=",",row.names=
      FALSE,col.names=(i==1),append=(i!=1))
    small.count <- sum(!is.na(match(names(ranking)[1:
      num.small],spike.names)))
    results <- data.frame(iteration=i,small.count)
    outfile <- paste("RVMFoldCountGLDilution.csv",sep="")
    write.table(results,file=outfile,sep=",",row.names=
      FALSE,col.names=(i==1),append=(i!=1))
    fold.change <- spike.conc[i+1,] / spike.conc[1,]
    fold.rank <- rank(fold.change,ties.method="min")
    fold.rank <- rep(length(unique(fold.rank)):1,times=
      as.vector(table(fold.rank)))
    outfile <- paste("RVMFoldRankGLDilution.csv",sep="")
    results <- data.frame(name=spike.names,iteration=rep(i,
      length(spike.names)),expectedrank=
      fold.rank[abs(spike.group[spike.names])],actualrank=
      ranking[spike.names])
    write.table(results,file=outfile,sep=",",row.names=
      FALSE,col.names=(i==1),append=(i!=1))

}

# End of GeneLogic Dilution Analysis
```

Appendix B
Quality Assessment Plots for All Datasets

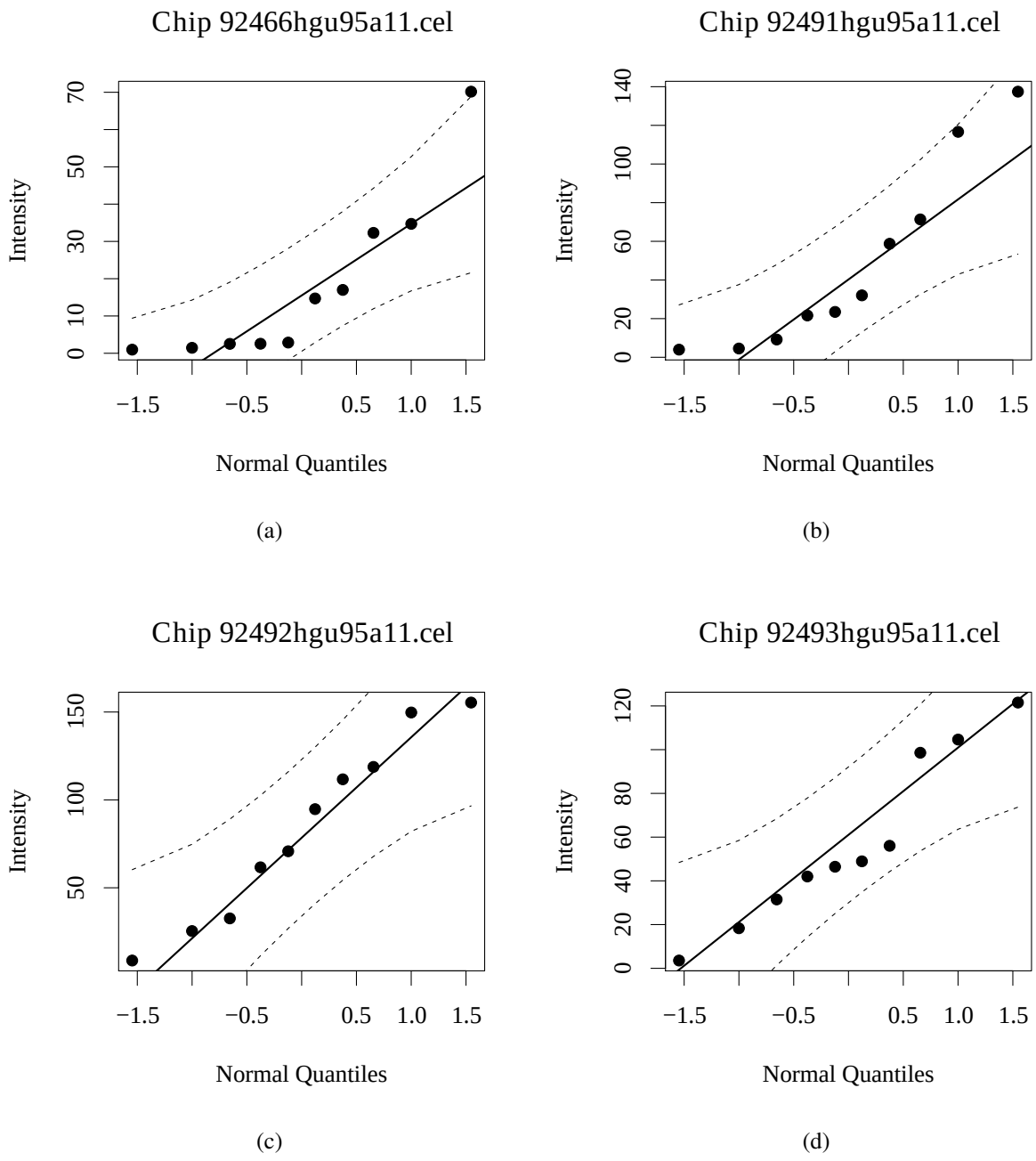

Figure B.1: GeneLogic Dilution Quality for Experiments 1–4. Plots of the computed MAS5 intensity values versus theoretical normal quantiles for a subset of chips. All intensity values are scaled to give a median intensity value of 100 for each chip.

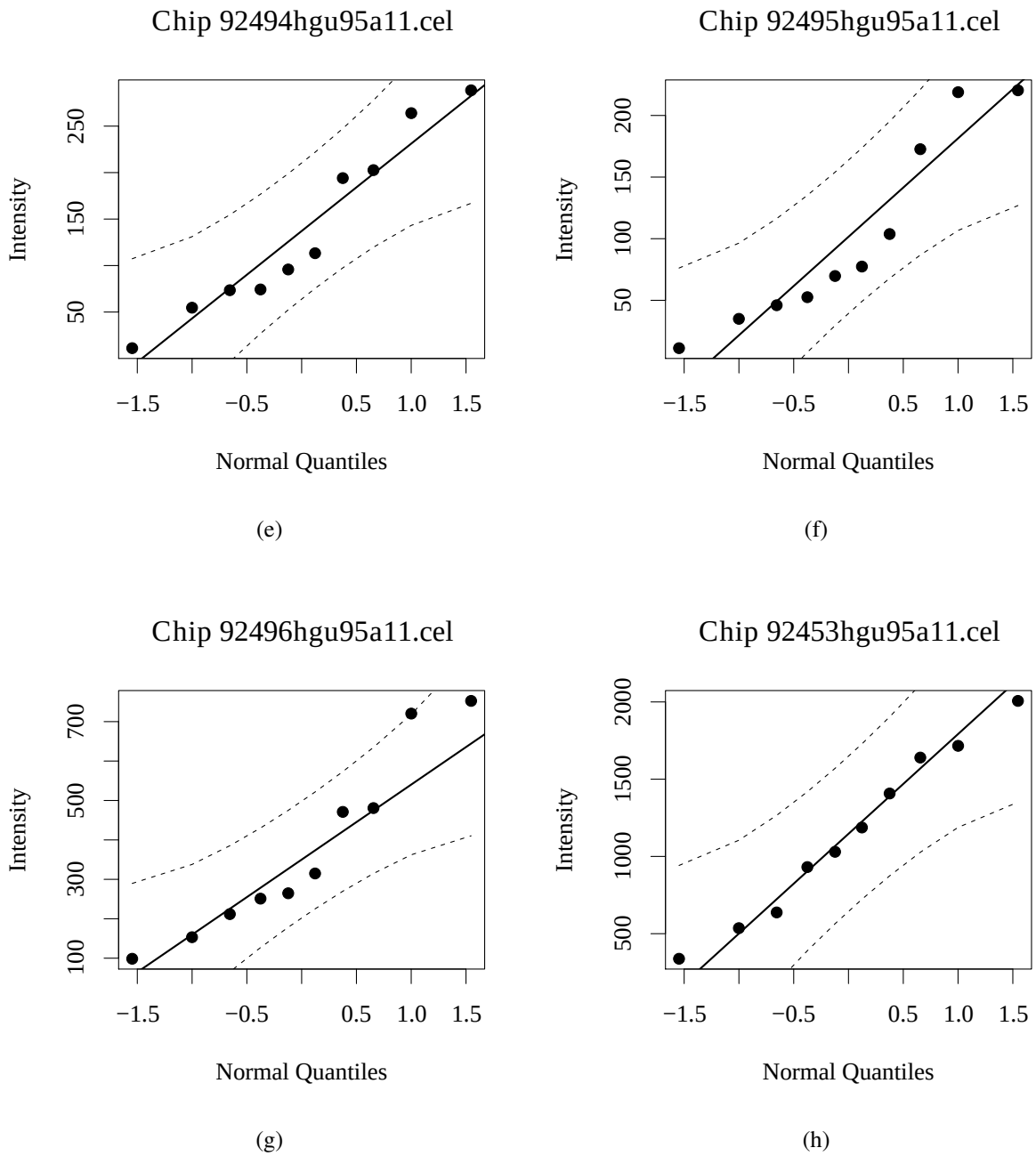

Figure B.1: GeneLogic Dilution Quality for Experiments 5–8.

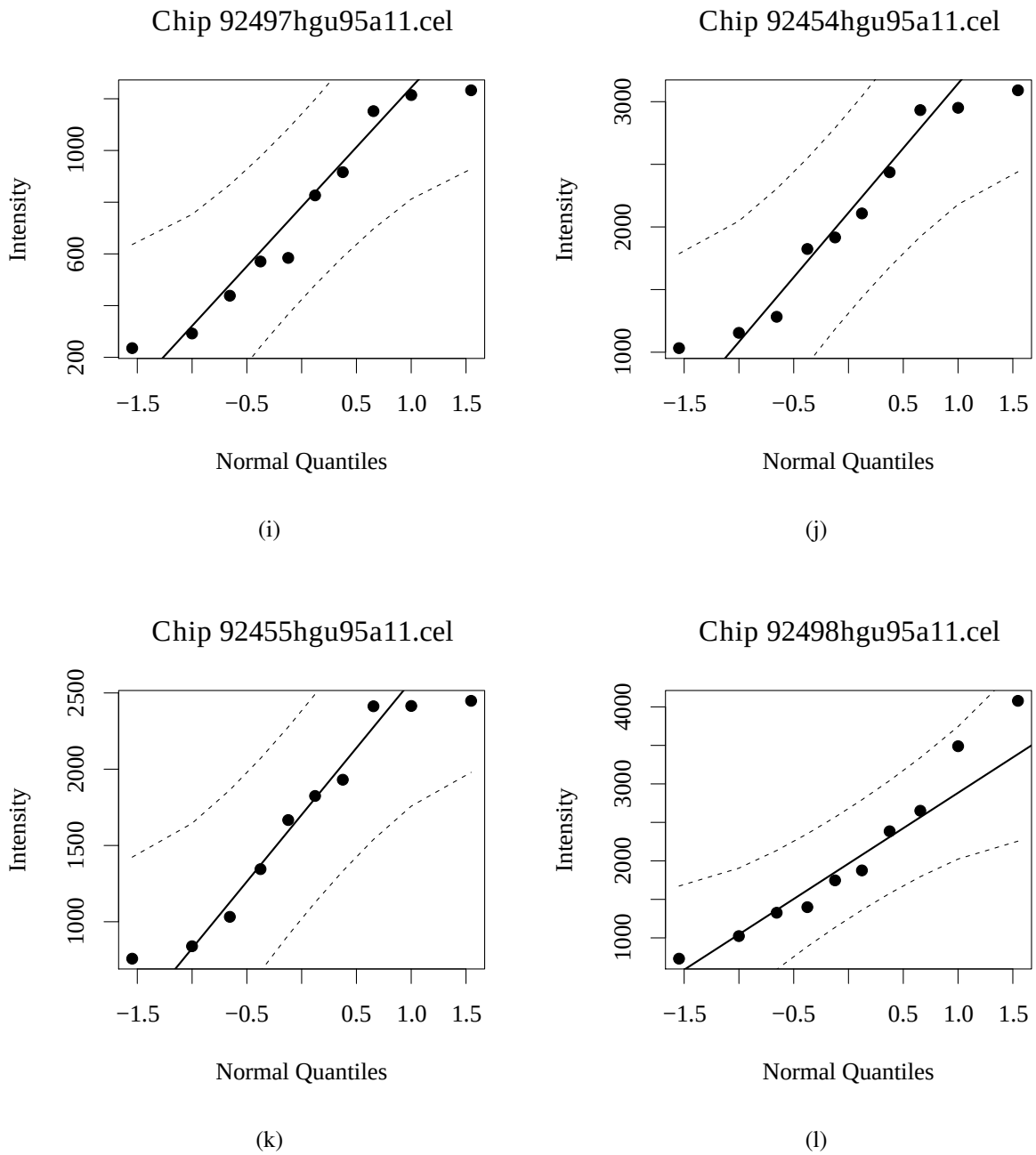

Figure B.1: GeneLogic Dilution Quality for Experiments 9–12.

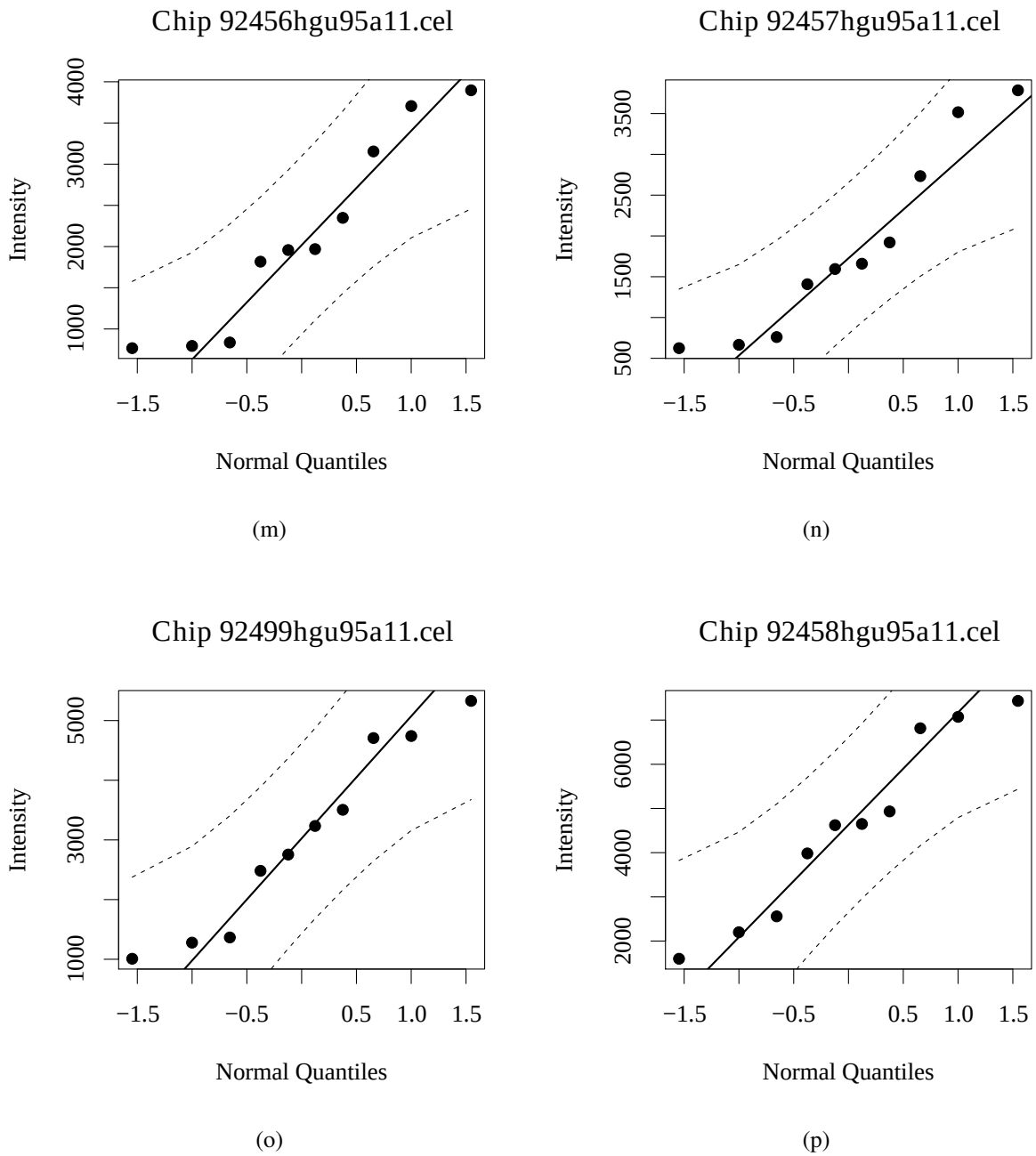

Figure B.1: GeneLogic Dilution Quality for Experiments 13–16.

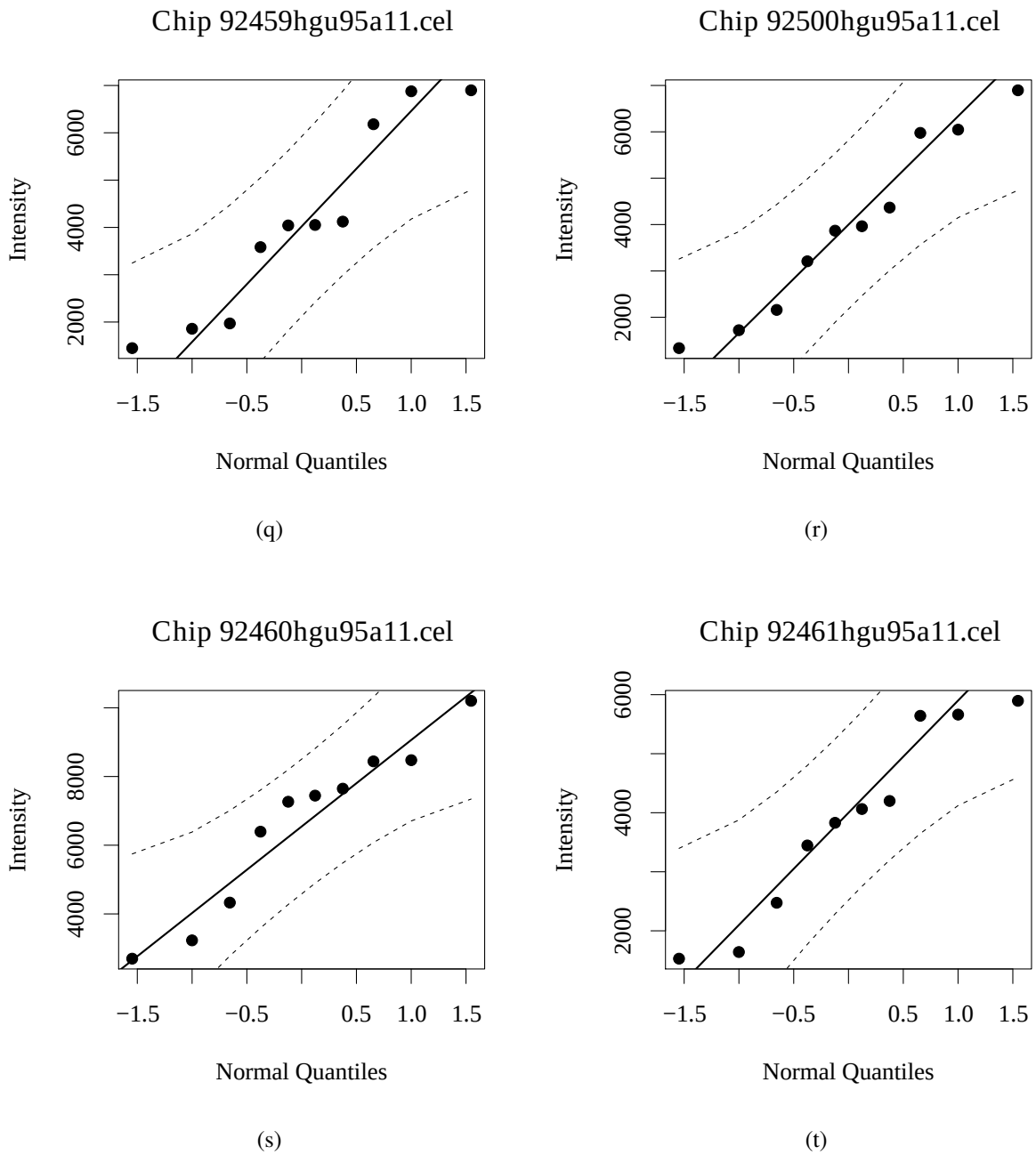

Figure B.1: GeneLogic Dilution Quality for Experiments 17–20.

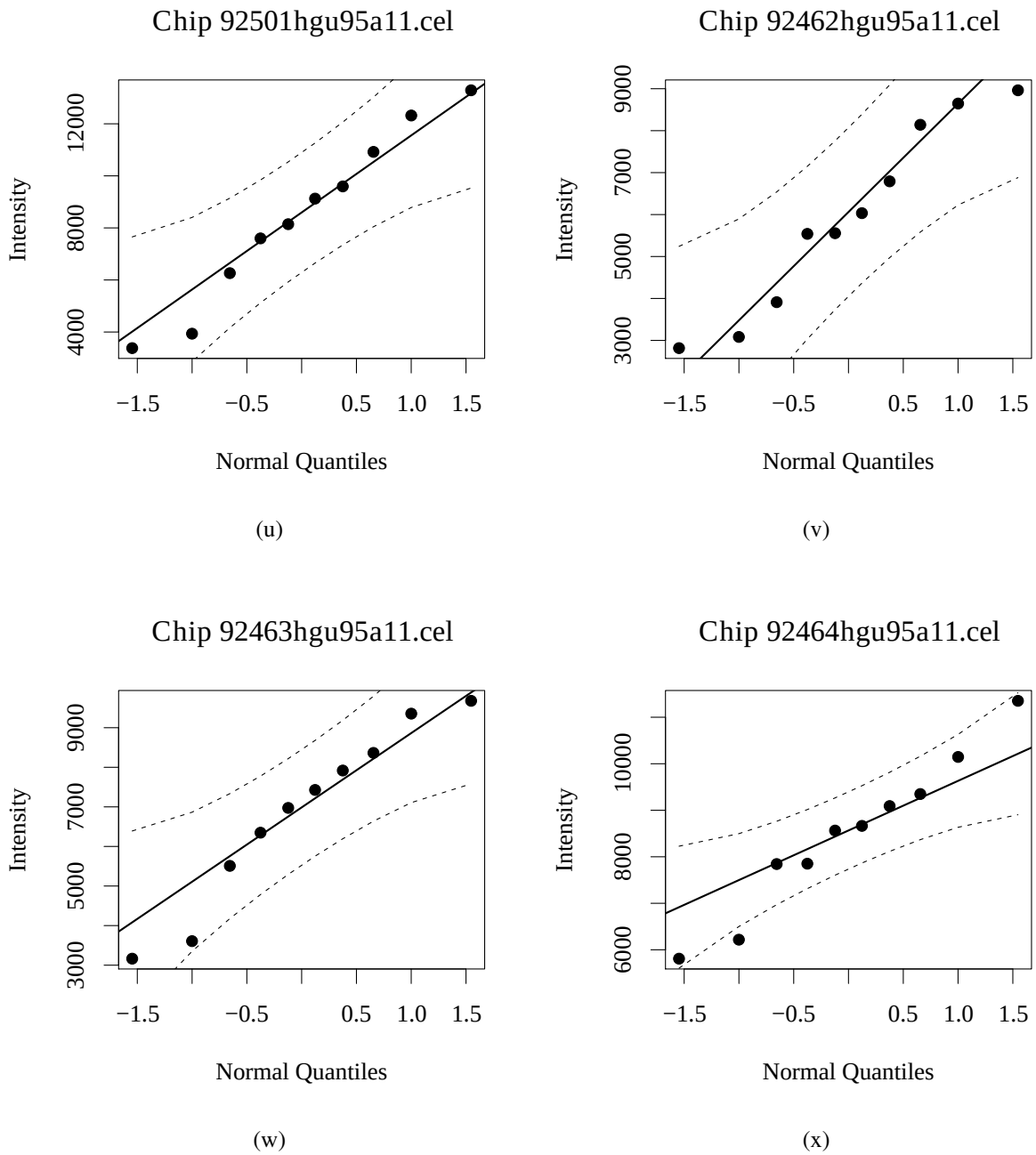

Figure B.1: GeneLogic Dilution Quality for Experiments 21–24.

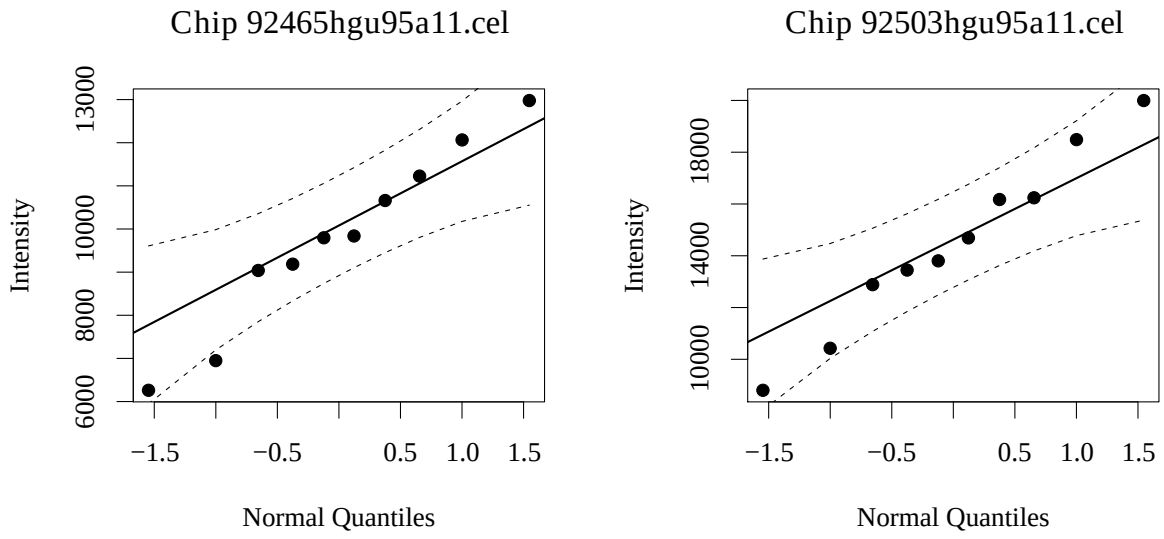

Figure B.1: GeneLogic Dilution Quality for Experiments 25–26.

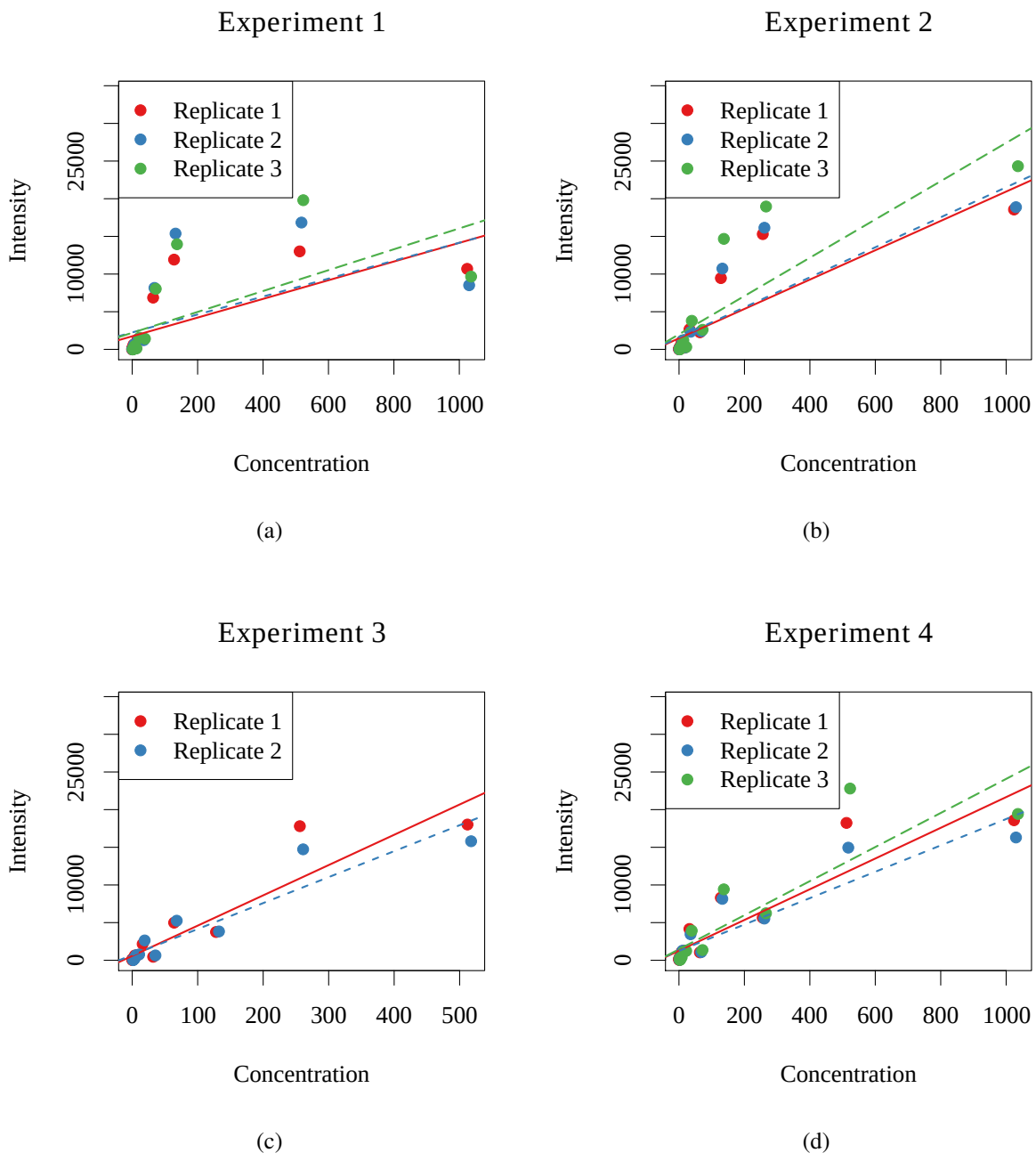

Figure B.2: Affymetrix U95 Latin Square Quality for Experiments 1–4. Plots of the computed MAS5 intensity values versus concentration for a subset of chips. High-quality chips would be expected to show a linear increase in intensity as concentration increases. All intensity values are scaled to give a median intensity value of 100 for each chip.

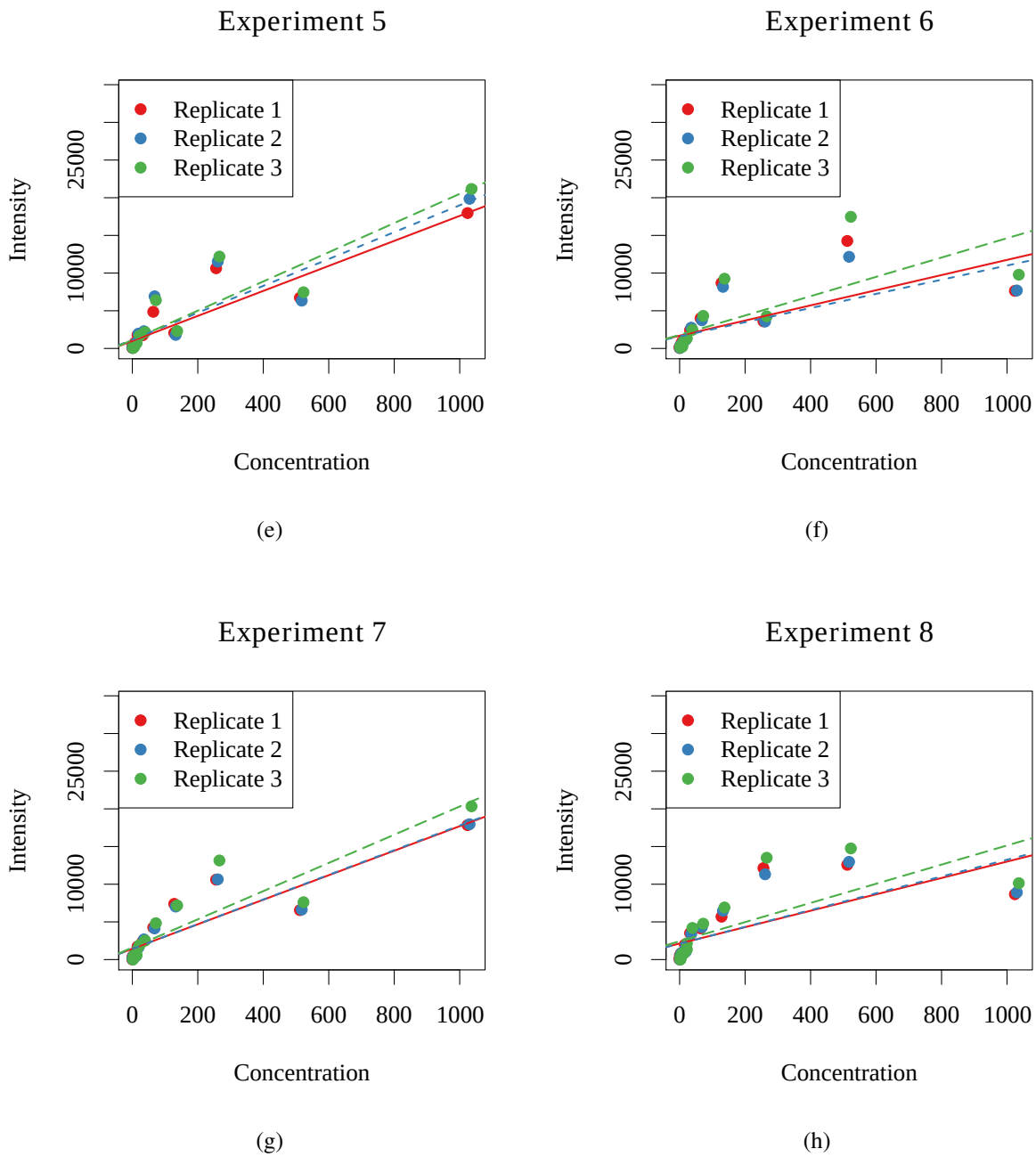

Figure B.2: Affymetrix U95 Latin Square Quality 5–8.

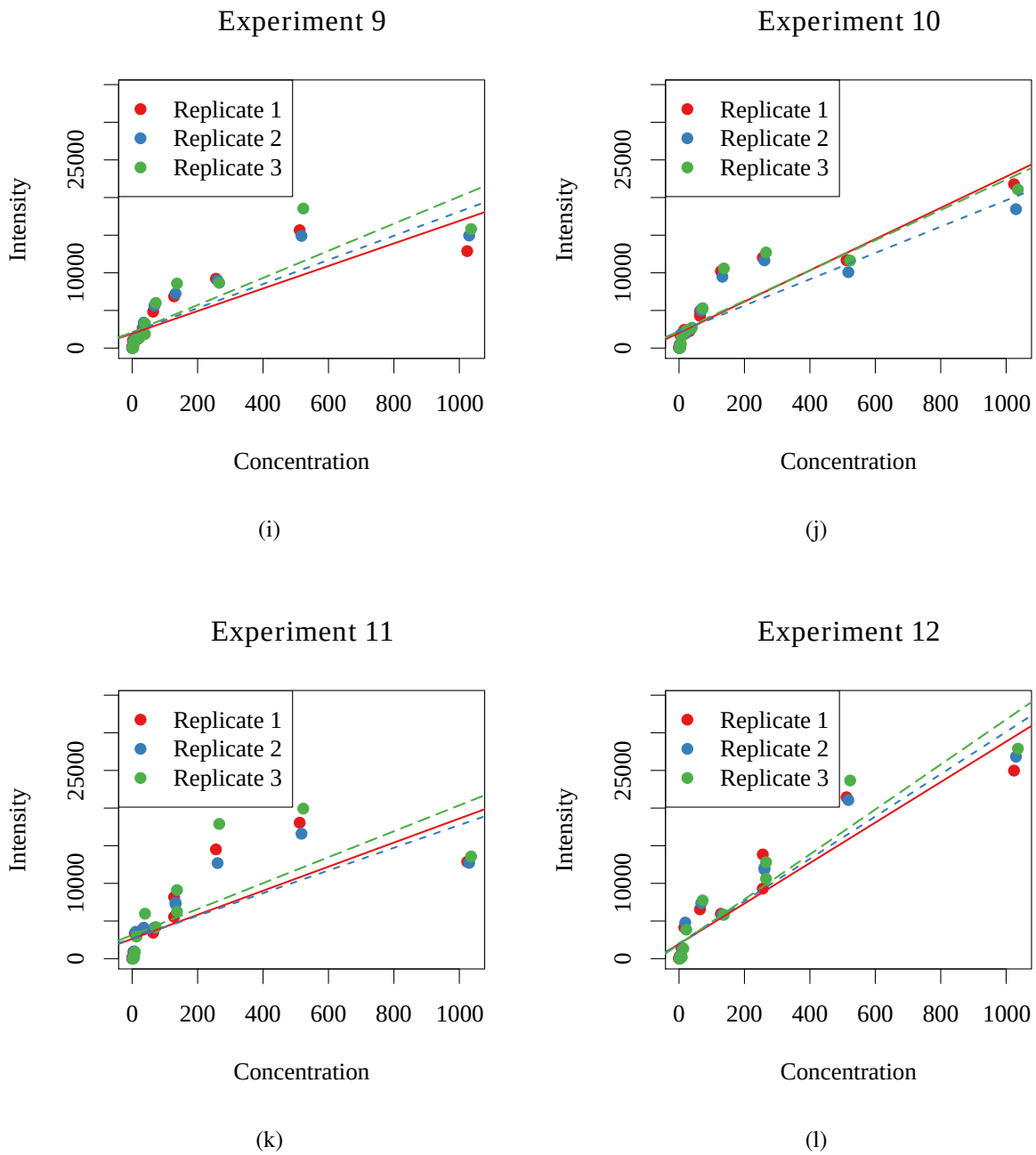

Figure B.2: Affymetrix U95 Latin Square Quality for Experiments 9-12.

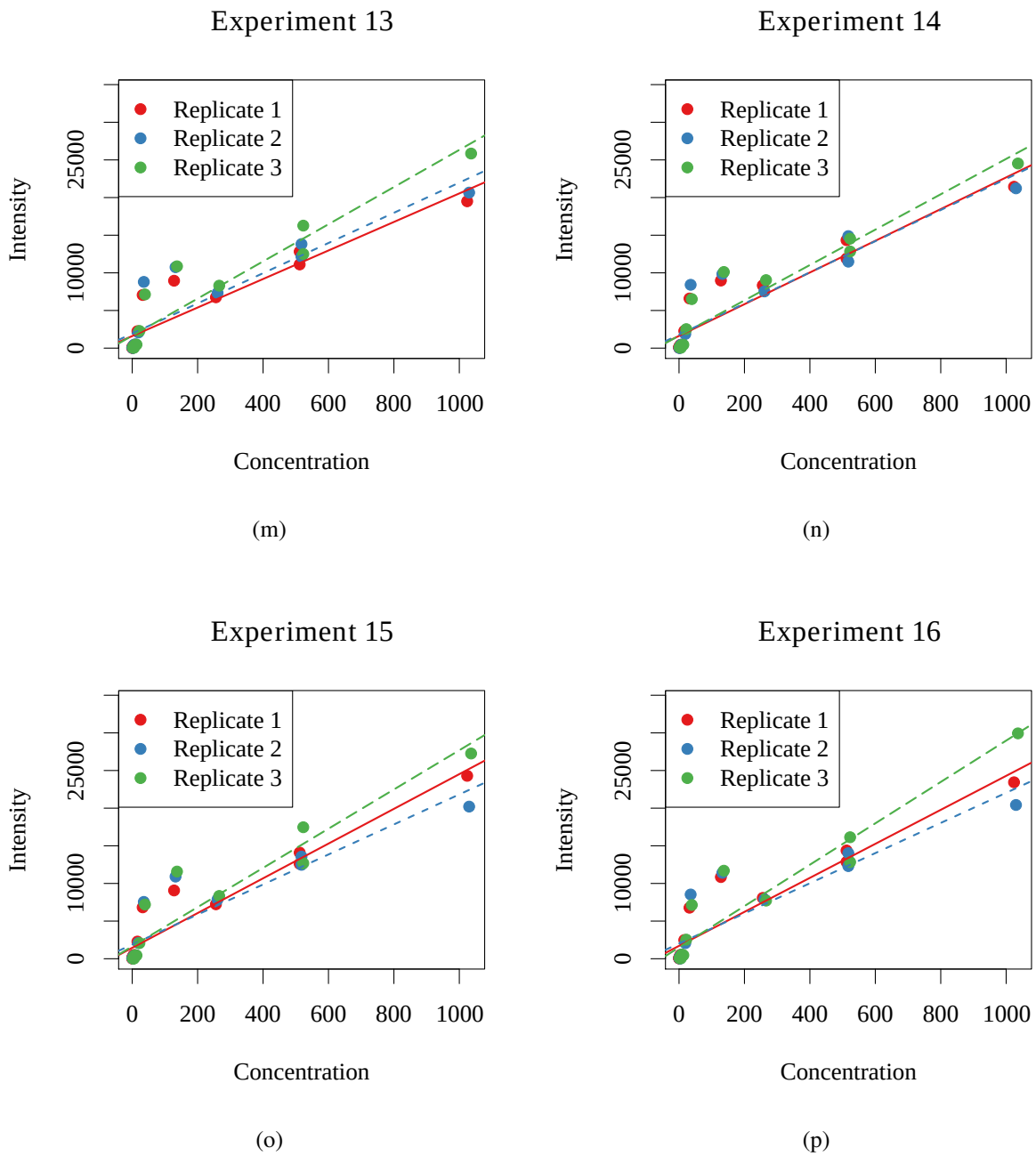

Figure B.2: Affymetrix U95 Latin Square Quality for Experiments 13–16.

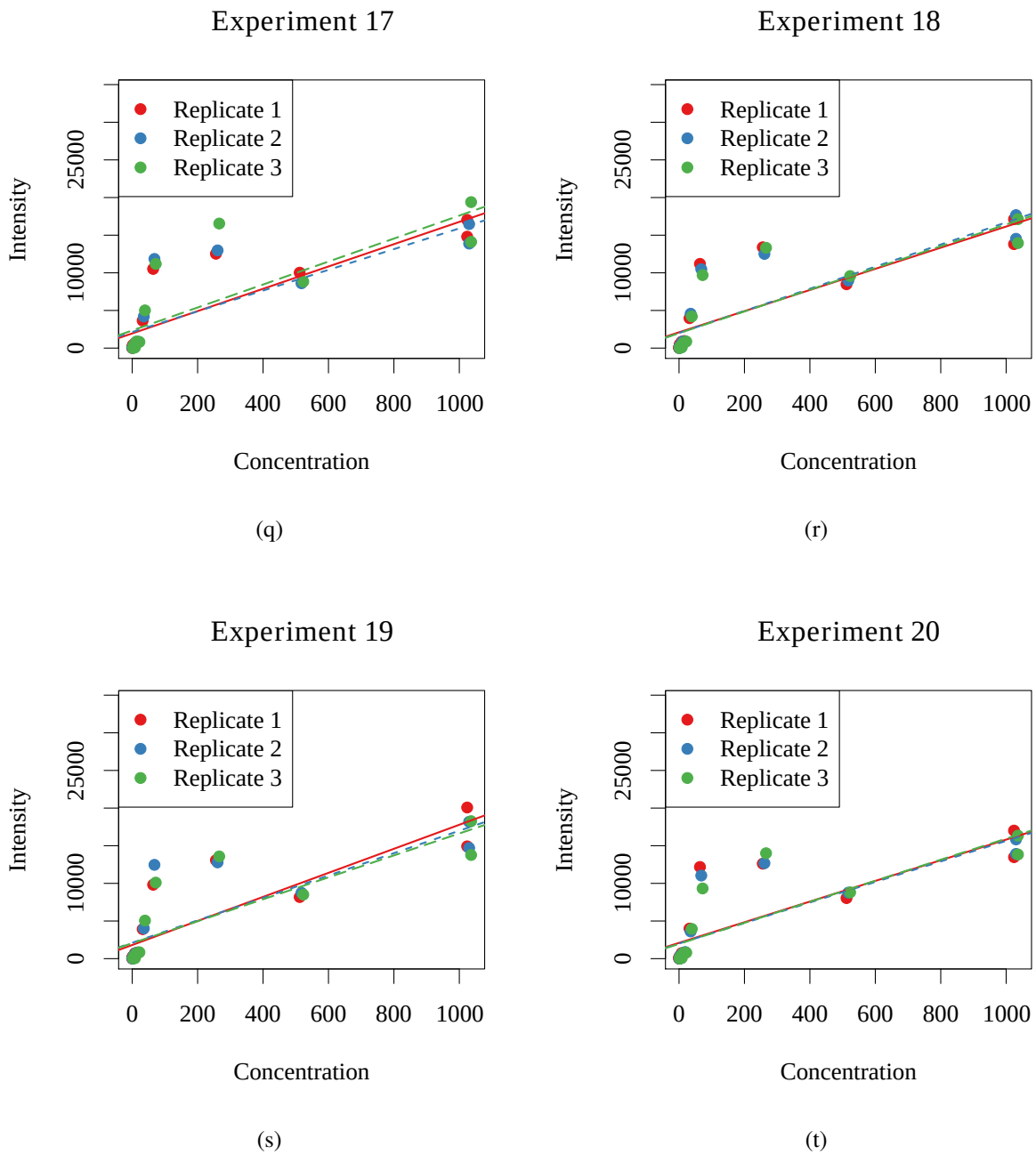

Figure B.2: Affymetrix U95 Latin Square Quality for Experiments 17–20.

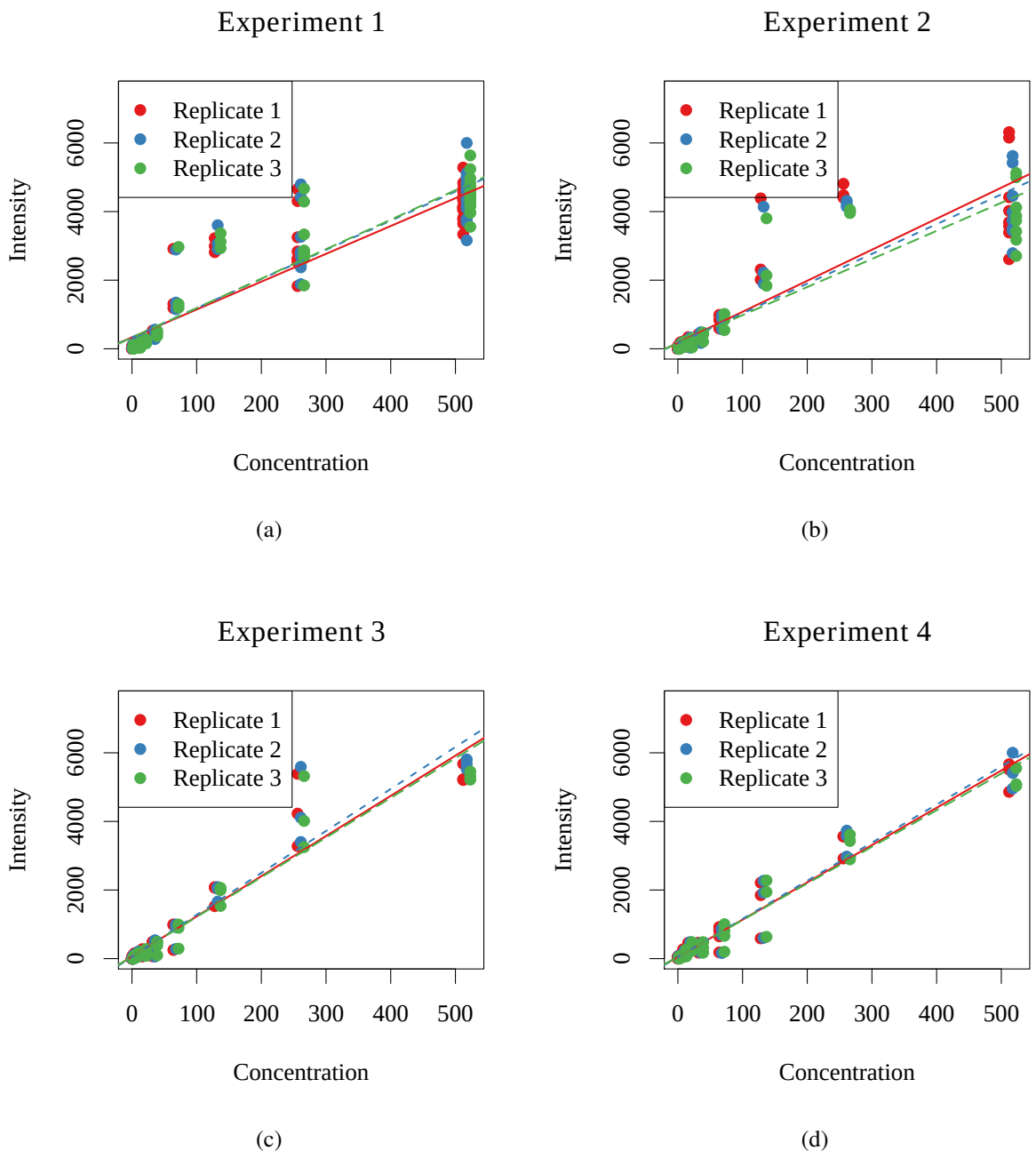

Figure B.3: Affymetrix U133 Latin Square Quality for Experiments 1–4. Plots of the computed MAS5 intensity values versus concentration for a subset of chips. High-quality chips would be expected to show a linear increase in intensity as concentration increases. All intensity values are scaled to give a median intensity value of 100 for each chip.

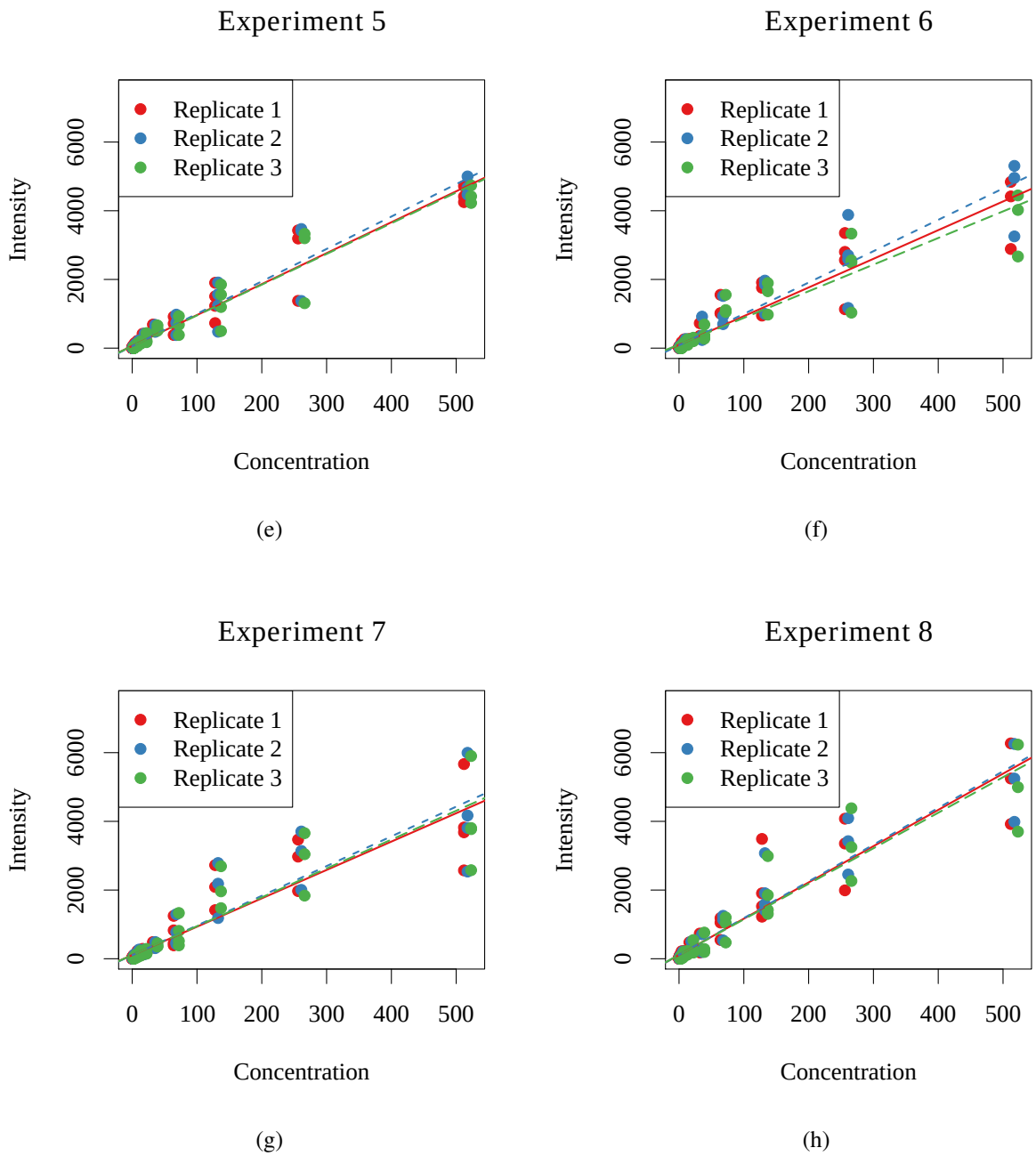

Figure B.3: Affymetrix U133 Latin Square Quality for Experiments 5–8.

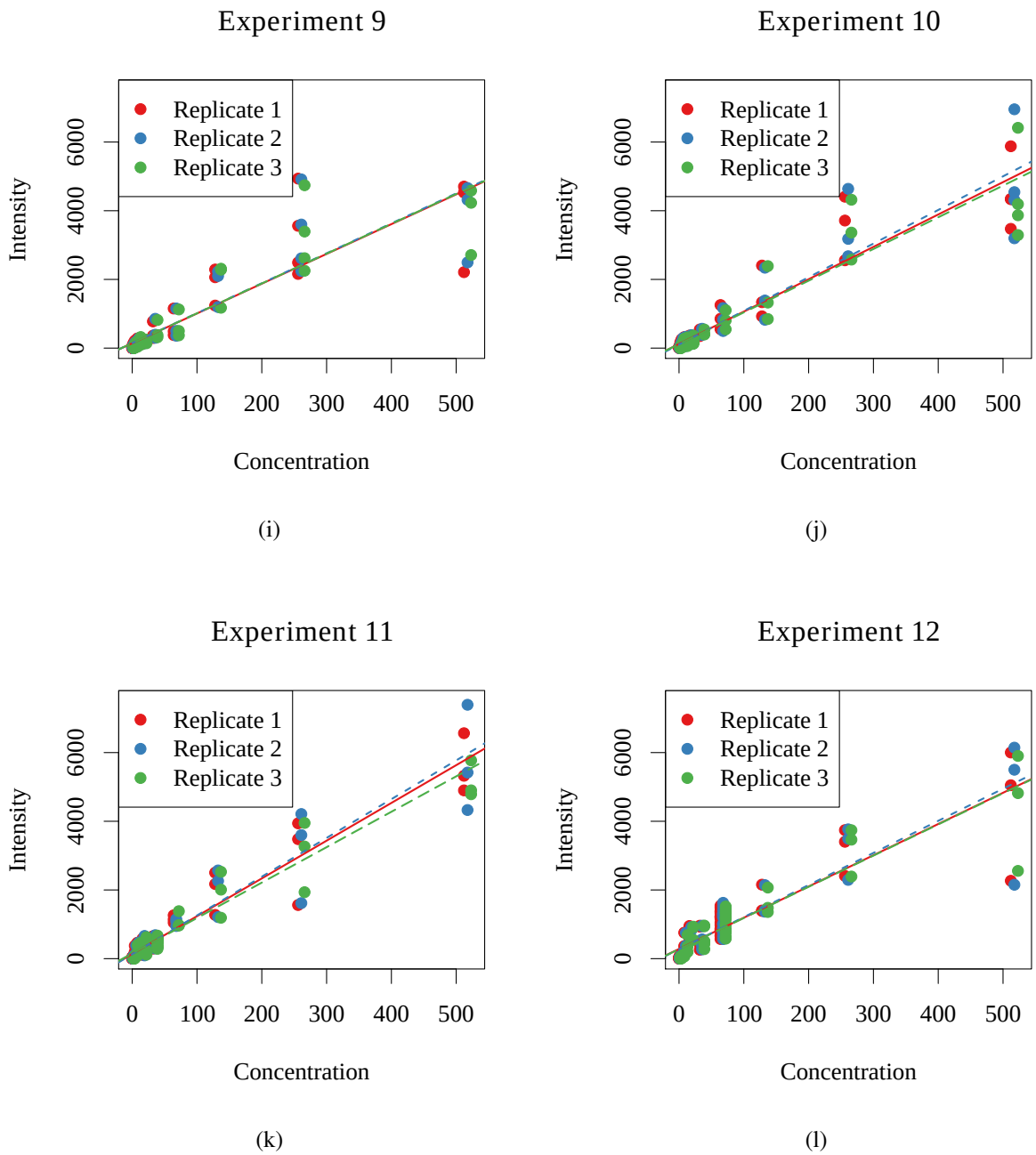

Figure B.3: Affymetrix U133 Latin Square Quality for Experiments 9–12.

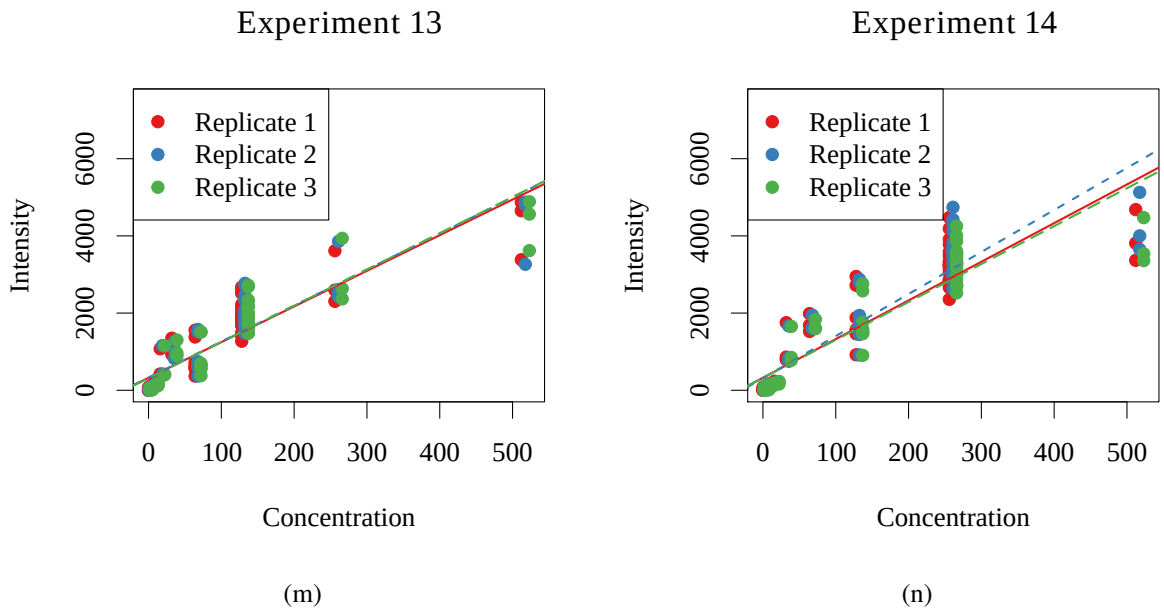

Figure B.3: Affymetrix U133 Latin Square Quality for Experiments 13–14.

Appendix C
Annotation Data for Omitted Probesets

Table C.1: Probesets Omitted from the Affymetrix U95 Latin Square and GeneLogic Dilution Analysis

Probeset Name	Number of Probes	Gene Symbol	GenBank ID	LocusLink ID	Annotation
1089_i_at	6	none	M64936	none	none
1152_i_at	6	CGB	J00117	1082	chorionic gonadotropin, beta polypeptide
1404_r_at	6	CCL5	M21121	6352	chemokine (C-C motif) ligand 5
1526_i_at	6	FGF8	U47011	2253	fibroblast growth factor 8 (androgen-induced)
1662_r_at	6	none	HG2261-HT2351	none	none
1672_f_at	6	RB1	L41913	5925	retinoblastoma 1 (including osteosarcoma)
1998_i_at	6	BAX	U19599	581	BCL2-associated X protein
2070_i_at	6	MAPK8	L26318	5599	mitogen-activated protein kinase 8
1944_f_at	7	MLH1	AF001359	4292	mutL homolog 1, colon cancer, nonpolyposis type 2 (E. coli)
2041_i_at	7	ABL1	M14752	25	v-abl Abelson murine leukemia viral oncogene homolog 1
796_i_at	7	CDKL1	X66358	8814	cyclin-dependent kinase-like 1 (CDC2-related kinase)
124_i_at	8	CDK10	X78342	8558	cyclin-dependent kinase (CDC2-like) 10
326_i_at	8	none	HG1800-HT1823	none	none
729_i_at	8	none	HG2147-HT2217	none	none
970_r_at	8	USP9X	X98296	8239	ubiquitin specific peptidase 9, X-linked
114_r_at	9	MAPT	X14474	4137	microtubule-associated protein tau
1280_i_at	9	none	HG2702-HT2798	none	none
219_i_at	9	MAP2	S76756	4133	microtubule-associated protein 2
850_r_at	9	IRS1	S62539	3667	insulin receptor substrate 1
411_i_at	10	IFITM2	X57351	10581	interferon induced transmembrane protein 2 (1-8D)

Probeset Name	Number of Probes	Gene Symbol	GenBank ID	LocusLink ID	Annotation
1553_r_at	11	CYP2A13	U22028	1553	cytochrome P450, family 2, subfamily A, polypeptide 13
1569_r_at	11	IL10RB	L42243	3588	interleukin 10 receptor, beta
2090_i_at	11	WNT6	H12458	7475	wingless-type MMTV integration site family, member 6
396_f_at	11	EPOR	X97671	2057	erythropoietin receptor
1245_i_at	12	PAK2	U25975	5062	p21 (CDKN1A)-activated kinase 2
1261_i_at	12	GSTA2	M16594	2939	glutathione S-transferase A2
1757_i_at	12	CYP3A7	D00408	1551	cytochrome P450, family 3, subfamily A, polypeptide 7
1758_r_at	12	CYP3A7	D00408	1551	cytochrome P450, family 3, subfamily A, polypeptide 7
32429_f_at	12	none	X68688	7558	none
35168_f_at	12	COL16A1	M92642	1307	collagen, type XVI, alpha 1
35175_f_at	12	EEF1A2	L10340	1917	eukaryotic translation elongation factor 1 alpha 2
37688_f_at	12	FCGR2A	M31932	2212	Fc fragment of IgG, low affinity IIa, receptor (CD32)
38831_f_at	12	GNB2	AF053356	2783	guanine nucleotide binding protein (G protein), beta polypeptide 2
39361_f_at	12	TSPAN6	AF043906	7105	tetraspanin 6
872_i_at	12	IRS1	S62539	3667	insulin receptor substrate 1
AFFX-hum_alu_at	69	MED6	U14573	10001	mediator of RNA polymerase II transcription, subunit 6 homolog (S. cerevisiae)

Table C.2: Probesets Omitted from the Affymetrix U95 Latin Square and GeneLogic Dilution Analysis

Table C.3: Probesets Omitted from the Affymetrix U133 Latin Square Analysis

Probeset Name	Number of Probes	Gene Symbol	GenBank ID	LocusLink ID	Annotation
205914_s_at	8	GRIN1	NM_007327	2902	glutamate receptor, ionotropic, N-methyl D-aspartate 1
214309_s_at	10	C21orf2	AI435747	755	chromosome 21 open reading frame 2
33579_i_at	13	GALR3	Z97630	8484	galanin receptor 3
34031_i_at	13	KRIT1	U90269	889	KRIT1,ankyrin repeat containing
37462_i_at	13	SF3A2	L21990	8175	splicing factor 3a, subunit 2, 66kDa
41386_i_at	13	JMJD3	AB002344	23135	jumonji domain containing 3
1405_i_at	14	CCL5	M21121	6352	chemokine (C-C motif) ligand 5
33322_i_at	14	SFN	X57348	2810	stratifin
65133_i_at	14	ZNHIT4	AI862454	83444	zinc finger,HIT type 4
91816_f_at	14	RKHD1	C18318	399664	ring finger and KH domain containing 1
48030_i_at	15	C5orf4	H93077	10826	chromosome 5 open reading frame 4
50314_i_at	15	C20orf27	AI761506	54976	chromosome 20 open reading frame 27
AFFX-hum_alu_at	69	none	AFFX-HUM_ALU	none	none