ESTIMATION OF VARIANCE COMPONENTS WITH MISSING DATA

By

Xiaowei Li

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
AT
DALHOUSIE UNIVERSITY
HALIFAX, NOVA SCOTIA
NOVEMBER 1995

ISBN 0-612-15859-4

Name XIAOWEI LI

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

statistics — SUBJECT TERM

0463 — SUBJECT CODE

UMI

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS
Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION
General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523
Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS
Language
- General 0679
- Ancient 0289
- Linguistics 0290
- Modern 0291

Literature
- General 0401
- Classical 0294
- Comparative 0295
- Medieval 0297
- Modern 0298
- African 0316
- American 0591
- Asian 0305
- Canadian (English) 0352
- Canadian (French) 0355
- English 0593
- Germanic 0311
- Latin American 0312
- Middle Eastern 0315
- Romance 0313
- Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY
Philosophy 0422
Religion
- General 0318
- Biblical Studies 0321
- Clergy 0319
- History of 0320
- Philosophy of 0322

Theology 0469

SOCIAL SCIENCES
American Studies 0323
Anthropology
- Archaeology 0324
- Cultural 0326
- Physical 0327

Business Administration
- General 0310
- Accounting 0272
- Banking 0770
- Management 0454
- Marketing 0338

Canadian Studies 0385
Economics
- General 0501
- Agricultural 0503
- Commerce-Business 0505
- Finance 0508
- History 0509
- Labor 0510
- Theory 0511

Folklore 0358
Geography 0366
Gerontology 0351
History
- General 0578
- Ancient 0579
- Medieval 0581
- Modern 0582
- Black 0328
- African 0331
- Asia, Australia and Oceania 0332
- Canadian 0334
- European 0335
- Latin American 0336
- Middle Eastern 0333
- United States 0337

History of Science 0585
Law 0398
Political Science
- General 0615
- International Law and Relations 0616
- Public Administration 0617

Recreation 0814
Social Work 0452
Sociology
- General 0626
- Criminology and Penology 0627
- Demography 0938
- Ethnic and Racial Studies 0631
- Individual and Family Studies 0628
- Industrial and Labor Relations 0629
- Public and Social Welfare 0630
- Social Structure and Development 0700
- Theory and Methods 0344

Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES
Agriculture
- General 0473
- Agronomy 0285
- Animal Culture and Nutrition 0475
- Animal Pathology 0476
- Food Science and Technology 0359
- Forestry and Wildlife 0478
- Plant Culture 0479
- Plant Pathology 0480
- Plant Physiology 0817
- Range Management 0777
- Wood Technology 0746

Biology
- General 0306
- Anatomy 0287
- Biostatistics 0308
- Botany 0309
- Cell 0379
- Ecology 0329
- Entomology 0353
- Genetics 0369
- Limnology 0793
- Microbiology 0410
- Molecular 0307
- Neuroscience 0317
- Oceanography 0416
- Physiology 0433
- Radiation 0821
- Veterinary Science 0778
- Zoology 0472

Biophysics
- General 0786
- Medical 0760

EARTH SCIENCES
Biogeochemistry 0425
Geochemistry 0996
Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleoecology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES
Environmental Sciences 0768
Health Sciences
- General 0566
- Audiology 0300
- Chemotherapy 0992
- Dentistry 0567
- Education 0350
- Hospital Management 0769
- Human Development 0758
- Immunology 0982
- Medicine and Surgery 0564
- Mental Health 0347
- Nursing 0569
- Nutrition 0570
- Obstetrics and Gynecology 0380
- Occupational Health and Therapy 0354
- Ophthalmology 0381
- Pathology 0571
- Pharmacology 0419
- Pharmacy 0572
- Physical Therapy 0382
- Public Health 0573
- Radiology 0574
- Recreation 0575
- Speech Pathology 0460
- Toxicology 0383

Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences
Chemistry
- General 0485
- Agricultural 0749
- Analytical 0486
- Biochemistry 0487
- Inorganic 0488
- Nuclear 0738
- Organic 0490
- Pharmaceutical 0491
- Physical 0494
- Polymer 0495
- Radiation 0754

Mathematics 0405
Physics
- General 0605
- Acoustics 0986
- Astronomy and Astrophysics 0606
- Atmospheric Science 0608
- Atomic 0748
- Electronics and Electricity 0607
- Elementary Particles and High Energy 0798
- Fluid and Plasma 0759
- Molecular 0609
- Nuclear 0610
- Optics 0752
- Radiation 0756
- Solid State 0611

Statistics 0463

Applied Sciences
Applied Mechanics 0346
Computer Science 0984

Engineering
- General 0537
- Aerospace 0538
- Agricultural 0539
- Automotive 0540
- Biomedical 0541
- Chemical 0542
- Civil 0543
- Electronics and Electrical 0544
- Heat and Thermodynamics 0348
- Hydraulic 0545
- Industrial 0546
- Marine 0547
- Materials Science 0794
- Mechanical 0548
- Metallurgy 0743
- Mining 0551
- Nuclear 0552
- Packaging 0549
- Petroleum 0765
- Sanitary and Municipal 0554
- System Science 0790

Geotechnology 0428
Operations Research 0796
Plastics Technology 0795
Textile Technology 0994

PSYCHOLOGY
General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

Contents

List of Tables

List of Figures

Abstract

A method to estimate variance components with missing data is presented. A typical application is in aquaculture genetics, in which breeding procedure may produce thousands of individuals. This method enables us to estimate genetic variance components when only a small proportion of individuals, those with extreme phenotypes, have been identified. In aquaculture populations the individuals available for measurement will often be selected, i.e. will come from the upper tail of a size-at-age distribution, or the lower tail of an age-at-maturity distribution etc.

Standard analysis of variance or maximum likelihood estimation cannot be used when missing data is not missing at random because of the biased nature of the estimates. In our model-based procedure a full likelihood function is defined, in which the missing information has been taken into account. This likelihood function is transformed into a computable function which is maximized to get the estimates. The computational methodology is outlined and a program is available.

This method is applied to simulated data and aquacultural data. The results obtained are significantly and uniformly more accurate than those obtained by any of the standard methods. Different issues concerning the method (such as the existence, uniqueness, confidence intervals, robust procedure, and random effects estimation) have been discussed in the thesis.

Acknowledgements

I wish to express my sincere thanks to my supervisor, Dr. Christopher Field, for his invaluable advice and continual encouragement during my graduate studies. I am particularly indebted to him for his excellent guidance and patience throughout the thesis work. Without his help and encouragement, the completion of this work would have been impossible.

I would also like to express my sincere thanks to Dr. Roger Doyle. The idea of the thesis was initiated by his very interesting fish breeding problem. He introduced me to quantitative genetics and has spent many hours discussing with me various issues related to the new ideas which I have explored in the thesis.

I would like to thank my external examiner, Dr. Jerald Lawless, for his invaluable suggestions and comments, Dr. David Hamilton, Dr. Rajendra Gupta and Dr. Bruce Smith for reading the manuscript and providing many excellent suggestions and stimulating comments. Their input has greatly improved the quality of my work.

My gratitude goes to Chiwen, for his technical assistance and support, and to my parents for their love and support.

Chapter 1

Introduction

Consider the general linear mixed model

$$Y = X\beta + Z\gamma + \epsilon, \tag{1.1}$$

where Y is an $n \times 1$ vector of observed responses, $X_{n\times p}$ and $Z_{n\times q}$ are known design matrices, β is a $p \times 1$ vector of fixed effects, γ is a $q \times 1$ unobservable vector of random effects assumed to be distributed as $N(\mu, \Sigma)$, and ϵ is an $n \times 1$ vector of error terms, distributed as $N(0, \sigma_0^2 I)$, and $cov(\gamma, \epsilon) = 0$. The mean and variance of Y can be written as

$$E(Y) = X\beta + Z\mu$$

and

$$V = Var(Y) = V(Z\gamma + \epsilon) = Z\Sigma Z^T + \sigma_0^2 I.$$

Although the form of V is often known, it usually contains unknown parameters.

Traditionally, this model's domain of application has included survey sampling (Yates and Zacopancy, 1935; Cochran, 1939), the analysis of designed experiments (Yates, 1940; Rao, 1947), genetics (Fairfield and Smith, 1936; Henderson, 1950) and industrial problems (Brownlee, 1953).

In the setting of interest for our problem, we assume that γ has the form $\gamma^T = (\gamma_1, \ldots, \gamma_c)$, where each γ_i is a $q_i \times 1$ vector distributed independently as $N(0, \sigma_i I_{q_i \times q_i})$

and $\sum_{i=1}^{c} q_i = q$, so that $Var(\gamma)$ is diagonal. Likewise, Z is partitioned as $Z = (Z_1, \ldots, Z_c)$ where Z_i is an $n \times q_i$ matrix, so the model 1.1 can be written as

$$Y = X\beta + \sum_{i=1}^{c} Z_i\gamma_i + \epsilon.$$

and also

$$V = \sum_{i=1}^{c} \sigma_i^2 Z_i (Z_i)^T + \sigma_0^2 I.$$

The variances $\sigma_1^2, \ldots, \sigma_c^2$, and σ_0^2 are called variance components.

Given Y, some of the usual problems associated with the model 1.1 are:

1. estimation of β, the fixed vector parameter,
2. prediction of γ, the vector of latent variables,
3. estimation of Σ (which is referred to as variance components estimation), the dispersion matrix of γ.

The estimation problem in the model 1.1 centers on V. The BLUP (Best Linear Unbiased Prediction) of β and γ can be found using

$$\hat{\beta} = [X'V^{-1}X]^{-1}X'V^{-1}Y,$$

$$\hat{\gamma} = E(\gamma \mid y, \hat{\beta}, V) = Var(\gamma)Z'V^{-1}(y - X\hat{\beta})$$

which Harville (1976) derived by extending the Gauss-Markov theorem to cover random effects.

Unfortunately, finding the BLUE requires knowledge of V which is rarely available. Currently, the best procedure available is to estimate V and then act as if the estimate is the real value of V. In other words, if V is estimated with $\hat{V}$ then β and γ are estimated with

$$\hat{\beta} = [X'\hat{V}^{-1}X]^{-1}X'\hat{V}^{-1}Y,$$

$$\hat{\gamma} = E(\gamma \mid y, \hat{\beta}, \hat{V}) = \hat{\Sigma}Z'\hat{V}^{-1}(y - X\hat{\beta}).$$

According to Christensen (1987), if $\hat{V}$ is close to V, the estimates $\hat{\beta}$ and $\hat{\gamma}$ should be close to the BLUP of β and γ.

Furthermore, estimation of V is also of interest in its own right. In quantitative genetics, the interest is in the variabilities of different genetic and environmental factors which are the variance components. For example, it is important for a breeder to know which traits have some degree of heritability if he wants to make improvement to his livestock. The heritability is the ratio of genotypic variance to total variance. For example, the variance-covariance matrix of a two-way nested model is defined as

$$\Sigma = \sigma_1^2 Z_1 Z_1' + \sigma_2^2 Z_2 Z_2' + \sigma_0^2 I.$$

The heritability is a function of variance components

$$h^2 = \frac{4\sigma_1^2}{\sigma_1^2 + \sigma_2^2 + \sigma_0^2}.$$

The literature on variance components is quite immense and there are various ways to estimate variance components, such as ANOVA (Analysis of Variance Estimators) or MLE (Maximum Likelihood Estimators). It has been common practice to estimate variance components by ANOVA for balanced data and by MLE for unbalanced data. But either approach requires the following to get good estimates:

- complete data,
- or data with values missing at random (in the sense that the observed units are a random subsample of the sampled units).

Statistical inferences are based only in part upon the observations. An equally important base is formed by prior assumptions about the underlying situation. The standard statistical methods are developed with an assumption, either implicit or explicit, that the process that caused the missing data can be ignored (Rubin, 1976).

If the missing data is nonignorable, analyses on the reduced sample that do not allow for this feature are subject to bias (Little and Rubin, 1987).

The following example provides some motivation.

A total of 1260 *Atlantic salmon* offspring were produced by a nested mating design (7 sires with 3 dams nested within each sire). It is supposed that the largest 200 were analyzed by DNA fingerprinting in the Gene Probe Lab at Dalhousie University so that their parentage is known. In fact, the true population values of mean and variances are known for 1260 simulated fish, but this would not be the case in an actual experiment. If the parentage of the remaining 1060 is considered to be unknown, the objective is to estimate the variance components of sires, dams and individuals with the 200 observations.

This is a two-way nested breeding experiment,

$$y_{ijk} = \mu + \alpha_i + \beta_{j(i)} + \epsilon_{k(ij)}, \begin{cases} i = 1, 2, \ldots, 7 \\ j = 1, 2, \ldots, 3 \\ k = 1, 2, \ldots, 60 \end{cases}$$

where

- μ is an unknown constant,
- $\alpha_i \sim N(0, \sigma_1^2)$,
- $\beta_{j(i)} \sim N(0, \sigma_2^2)$,
- and $\epsilon_{k(ij)} \sim N(0, \sigma_0^2)$.

Analyzing the 200 data by standard ANOVA and MLE produce severe biases as shown in Table 1.1. The estimates of σ_1^2 and σ_2^2 as shown in lines 2 and 3 of the table underestimate the estimates based on complete sample by factors of 10 or more in some case.

Table 1.1: *Estimates by two standard methods compared with estimates based on complete sample*

	σ_0^2	σ_1^2	σ_2^2	μ
True(1260)	100.75	29.05	18.70	28.17
ANOVA(200)	26.43	2.69	-1.019	47.10
MLE(200)	25.83	1.89	0.082	46.30

The natural questions concerning this example becomes how to use the data observed and the partial information on the missing data (i.e. the fact that only largest 200 are observed) to get sensible estimates.

There are other setting where missing data occurs and missing data are not missing randomly, the same questions need to be asked. Here are some examples

- the respondents in a household survey may refuse to report income when income is the variable of interest
- in an industrial experiment some results are missing because of mechanical breakdowns
- some patients will survive to the end of a clinical trial
- in animal breeding procedure, selection typically occurs by size grading during grow-out and/or choice of superior ones as brood stock

Typical fish genetic experiments involve hundreds of thousand of individuals, only a small proportion of which will be identified by DNA fingerprinting. These fish will typically be selected, e.g. will come from the upper tail of a size-at-age distribution, or the lower tail of an age-at-maturity distribution etc. Estimation of genetic and environmental variance components using DNA fingerprint pedigrees will therefore involve a high proportion of missing and selected data. We will not have a complete data set in this situation.

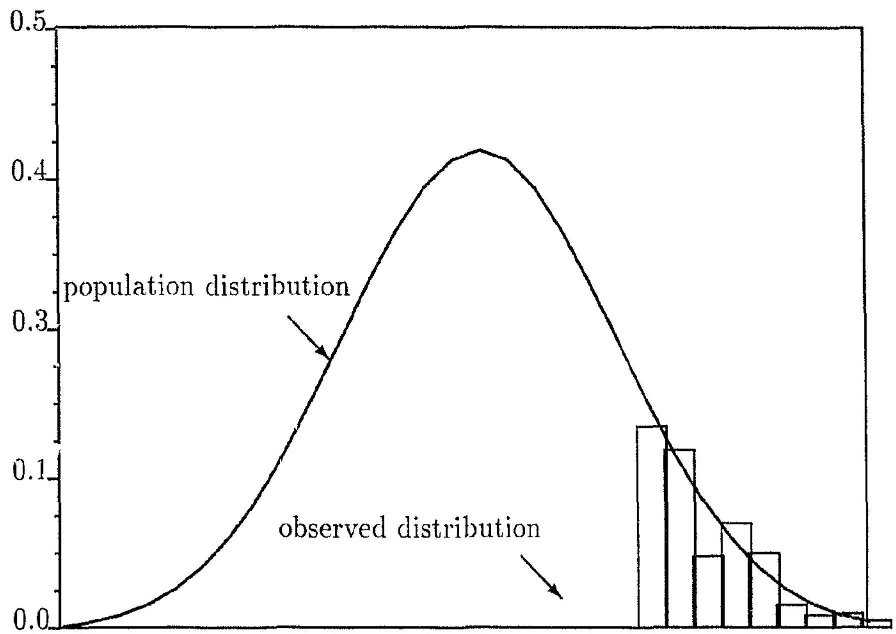

Figure 1.1: Observed and population distributions. The blank area under curve represents missing data.

Since the probability of y_{ijk} observed depends on the value of y_{ijk}, the missing data is not missing at random. Our data came from a relatively small proportion of the population. The histogram in Figure 1.1 illustrates an observed distribution and population distribution.

The ANOVA and MLE estimation in above example produced severe biases because of ignoring missing data which are not missing randomly (each of 1060 non sampled offspring is smaller than any of the sampled offspring). This indicates that we can not ignore this pattern of missing data.

Finding a method to analyses such data is the focus of this thesis. A method is presented to estimate the variance components with high proportion of missing data both for one-way and two-way nested models. In our model-based procedure, a full likelihood function is defined in which the information about missing data has been taken into account. This function is transformed into a computable function which can be maximized to get the estimates.

The mechanisms that lead to missing data is a key element in choosing an appropriate analysis and in interpreting the results. Sometimes the mechanism is under the control of the statistician. The case of *censoring* is a situation where the mechanism leading to missing data may not be under the control of the statistician, but is understood. The data consist of times to the occurrence of an event (e.g., death of an experimental animal, birth of a child, failure of a light bulb). For some units in the sample, time to occurrence is censored because the event had not occurred before the termination of the experment. If the time to censoring is recorded, then we have the partial information that the failure time exceeds the time to censoring. The analysis of the data needs to take account of this information to avoid biased results. In Type I censoring, the cause of censoring is the planned ending of follow-up at a predetermined time. In Type II censoring, observation ceases after a predetermined number of failures. The type of censoring handled in the thesis is Type I censoring (we consider the fixed cutoff). This is to keep the problem manageable. In our motivating example, the type of censoring is Type II censoring (the cutoff is random). Random cutoff will be considered in future work.

In Chapter 2, several methods of variance components estimation are reviewed for both point and interval estimates. Methods for doing statistical analysis with missing data are also listed in this chapter. Chapter 3 and Chapter 4 present the new methods for one-way model and two-way nested model respectively. Much of the chapters is devoted to the details of the full likelihood function and the results of the estimates. The theoretical basis of the present method is organized in Chapter 5. To transform the full likelihood function to a computable function is the key step in developing the method and this transformation is described in chapter 5. The existence and uniqueness of the maximum of the transformed function are also crucial for the new technique. Chapter 5 includes these proofs. The confidence intervals of variance components with missing data are also constructed in Chapter 5. Robust procedures for the one-way model are proposed in Chapter 6. Finally, Chapter 7 discusses the estimation of random effects.

The new method is applied to both aquicultural examples and simulated data sets showing that our estimates are significantly and uniformly more accurate than those obtained by any of the standard procedures.

Chapter 2

Review

The problem of estimation of variance components in random and mixed linear models has received much attention in the statistics literature, as for instance in Khuri and Sahai (1985). There are several approaches to this problem, such as the analysis of variance (ANOVA) estimator (reviewed by Searle, 1971), and the maximum likelihood estimator (MLE) (Hartley and Rao, 1967). It has been common practice to estimate the variance components by ANOVA for balanced data and by MLE for unbalanced data.

The ANOVA estimates are obtained by equating observed and expected mean squares in the analysis and solving the resulting equation for the estimators. These estimators are unbiased and can be expressed as quadratic functions of the observations. The main desirable feature of these estimators is their simple computation. Under normality and balanced data, they have minimum variance among all unbiased estimators (Graybill, 1954). However they can yield negative estimates and even under normality assumptions their distributions are intractable. For unbalanced data, the choice of appropriate quadratic forms poses a difficult problem. The estimates obtained may be not unbiased.

Another approach to variance components estimation is that of maximum likelihood. The maximum likelihood approach is based on assuming density of the data and then maximizing the likelihood function over the parameter space under nonnegative constraints on the variance components. The maximum likelihood estimators are a function of the sufficient statistics, are consistent and are asymptotically normal and efficient (Harville, 1977). In particular, the maximum likelihood approach is "always" well-defined since nonnegative constraints on the variance components or other constraints on the parameter space or incompleteness in the data cause no conceptual difficulties. In spite of their good statistical properties, maximum likelihood estimators of variance components have not been used much in practice. The most important reason for this is the computation of the ML estimate requires the numerical solution of a constrained nonlinear optimization problem. Prior to the advent of the computer, this requirement presented a virtually insurmountable barrier to their use. Even after computer became commonplace, a constrained nonlinear optimization problem is, in general, a difficult numerical problem. The maxima can occur on the boundary of the parameter space and the log likelihood surface can have local maxima. Unfortunately, no known techniques guarantee convergence to a global maximum from arbitrary starting values.

We proceed with the ML estimates for variance components as follows. For balanced data or unbalanced data in model 1.1, we assume that $Y_{n\times 1}$ is multivariate normal and that V is nonsingular, so that the density of Y exists and is given by

$$(2\pi)^{-n/2}\det(V)^{-1/2}\exp[-\frac{1}{2}(Y-X\beta)'V^{-1}(Y-X\beta)].$$

The log likelihood is

$$\log L(\beta,V)=-\frac{n}{2}\log(2\pi)-\frac{1}{2}\log[\det(V)]-\frac{1}{2}(Y-X\beta)'V^{-1}(Y-X\beta) \qquad (2.1)$$

By definition, maximum likelihood estimates $\hat{\beta}$ and $\hat{V}$ are values satisfying

$$L(\hat{\beta},\hat{V};Y)=max_{(\beta,V)}L(\beta,V;Y).$$

To obtain ML estimates, the usual approaches are either to maximize the likelihood function directly or to solve the first order equations. Explicit solutions to

ML equations are available only in special cases. In general one has to use iterative method to get solutions.

Several papers evaluate algorithms for variance components estimation (Dempster *et al*, 1984; Jennrich and Schlucher, 1986; Laird *et al*, 1988; Lindstrom and Bates, 1988). While is no consensus on the best method, some general conclusions seem to be as follows.

- The Newton-Raphson method often converges in the fewest iterations, followed by scoring method and then the EM algorithm. In some cases the EM algorithm requires a very large number of iterations.
- The robustness of the methods to their starting values (ability to converge given poor starting values) is the reverse of the rate of convergence.
- The EM algorithm automatically takes care of inequality constraints imposed by the parameter space. Other algorithm need specialized programming to incorporate constrains.

One criticism of the ML approach to the estimation of variance component is that the MLE takes no account of the loss in degrees of freedom that results from estimating β. A modification due to Patterson and Thompson (1971) is known as restricted maximum likelihood estimate (REML). REML finds maximum likelihood estimates from the distribution of the residuals. In other words, REML maximizes the part of the likelihood which is said to be location invariant. Harville (1974) showed that REML is equivalent to marginalizing the likelihood over the fixed effect parameters. For example, if we take $X\beta = \mu$, the REML can be obtained by maximizing the marginal likelihood

$$L_1(\sigma_0^2, \dots, \sigma_c^2 \mid Y) = \int L(\mu, \sigma_0^2, \dots, \sigma_c^2 \mid Y) d\mu.$$

In general, a REML is the values of $(\sigma_0^2, \dots, \sigma_c^2)$ that maximizes L_1.

MLE provides estimators of fixed effects, whereas REML does not. But with balanced data REML solutions are identical to ANOVA estimators which have optimal minimum variance properties. How does the REML compare with the MLE with regard to mean squared error (MSE)? In general, the answer depends on the specifics of the underlying model. For ordinary fixed ANOVA or regression models the MLE has uniformly smaller MSE then the REML when $rank(X) \leq 4$; however, the REML has smaller MSE when $rank(X) \geq 5$ and $n - rank(X)$ is sufficiently large. MSE comparisons between MLE and REML were made by Corbeil and Searle (1976) and by Hocking and Kutner (1975) for several mixed and random ANOVA model.

Besides ANOVA, MLE and REML, there are Bayesian methods (Hill, 1965, 1967), minimum norm quadratic unbiased estimate (MINQUE) (Rao, 1970, 1972, 1971, 1979), and minimum variance quadratic unbiased estimate (MIVQUE).

Several comparative studies on variance component estimators have been made. The main criterion for comparison was MSE, and the model used was the 1 way random model (Townsend and Searle, 1971; Swallow and Monahan, 1984), the 2 way crossed classification mixed model, or the 2-way nested random model (Corbeil and Searle, 1976). In Corbeil and Searle's paper, a comparison was made between MLE, REML, and ANOVA. On the basis of this comparison, MLE was favored. A comparison of ANOVA and MINQUE for the 1-way was made by Ahrens *et al* (1981). They determined that ANOVA was favored. Swallow and Monahan (1984) have made a Monte Carlo comparison of five estimators for 1-way model. The five estimators, namely ANOVA, MLE, REML, MIVQUE(0) and MIVQUE(A), were compared through their MSE, estimated by Monte Carlo simulation. Their results indicate that unless the data are severely unbalanced and $\sigma_1^2/\sigma_0^2 > 1$, the ANOVA are adequate. The MLE is preferred when $\sigma_1^2/\sigma_0^2 < 0.5$

Wald (1940, 1941) was the first to obtain exact confidence intervals on the ratio of two variance components for 1-way and 2-way crossed classification model without interaction. Burdick and Graybill (1984) developed a procedure for obtaining exact confidence intervals on certain positive linear combinations of the variance components for the 1-way random model. In general, approximate procedures have been used more frequently. Several methods (mostly approximate) for finding confidence intervals are reviewed in Burdick and Graybill (1988).

The literature on the analysis of incomplete data is comparatively recent. Methods proposed in this literature can be roughly grouped into the following categories (Little and Rubin, 1987):

- Procedures Based on Completely Recorded Units. In this approach the incompletely recorded units are discarded and only the units with complete data are analyzed. It is generally easy to carry out and may be satisfactory with small amounts of missing data but can lead to serious bias in other cases;

- Imputation-Based Procedures. The missing values are estimated and the resultant completed data are analyzed by standard method. Commonly used procedures for imputation include mean imputation, hot deck imputation, regression imputation and so on;

- Weighting Procedures. Let y_i be the value of a variable Y for unit i in the population. Then the population mean is often estimated by

$$\sum(\pi_i)^{-1}y_i/\sum(\pi_i)^{-1}$$

where the sums are over sampled units, π_i is the probability of inclusion in the sample for unit i and $(\pi_i)^{-1}$ is the design weight for unit i. Weighting procedures are then used modify the weight in an attempt to adjust for nonresponse.

- Model-Based Procedures. Define a model for the partially missing data and base inferences on the likelihood under that model, with parameters estimated by procedures such as maximum likelihood.

There is extensive literature for multivariate normal models with incomplete observations, including Wilks (1932), Anderson (1957), Afifi and Elashoff (1966), Hartley and Hocking (1971). For generalized linear models (GLM's), Chen and Fienberg (1974) discussed parameter estimation for two-dimensional contingency table with partially cross-classified observations. Fuchs (1982) analyzed the problem of incomplete data in log-linear models. Shafer (1987) examined the covariate measurement error in GLM's. Ibrahim (1990) worked on the problem of incomplete data for any GLM with discrete covariates, in which incompleteness is due to partially missing covariates on some observations.

The EM algorithm is a very general iterative algorithm for ML estimation in incomplete-data problems. Each iteration consists of two steps: the E-step (expectation step) and the M-step (maximization step). Formally, let θ^i denote the current guess to the mode of the observed likelihood $P(\theta \mid y_{obs})$, let $P(\theta \mid y_{obs}, y_{mis})$ denote the augmented posterior, and let $P(y_{mis} \mid \theta^i, y_{obs})$ denote the conditional predictive distribution of the unobserved data y_{mis}. The E-step consists of computing

$$Q(\theta \mid \theta^i) = \int \log P(\theta \mid y_{obs}, y_{mis}) P(y_{mis} \mid \theta^i, y_{obs}) dy_{mis}.$$

i.e. the expectation of $\log P(\theta \mid y_{obs}, y_{mis})$ with respect to $P(y_{mis} \mid \theta^i, y_{obs})$. In the M-step the Q function is maximized with respect to θ to obtain θ^{i+1}. The algorithm is iterated until $||\theta^{i+1} - \theta^i||$ is sufficiently small.

The EM algorithm has been used to obtain ML estimates of variance components and more generally covariance components (Dempster *et al*, 1977; Dempster *et al*, 1981). They treated the unobserved random variables (random effects) as missing data (with all Y observed) and use EM algorithm to obtain ML estimates.

Despite recent advances in the analysis of data with missing values, very little work has been done on variance components estimation with missing data. The major difficulty of this subject is that the observations are not independent. We can not write the full likelihood as we usually do in survival analysis

$$lik = \prod_{obs} f(t;\phi) \prod_{mis} F(c;\phi).$$

It will be also hard to apply EM algorithm to this subject because $P(\theta \mid y_{obs}, y_{mis})$ can not be written as linear in the unobserved data y_{mis} (Little and Rubin pointed out that estimates can be severely biased when EM approach is applied in general, 1983). Chapter 3 (section 3.5) gives more details about the difficulty of using EM algorithm to estimate variance components with incomplete Y. In this thesis, new methods of point estimates and approximate confidence intervals of variance components with a high proportion of data missing have been derived in the situation where the type of missing data is censoring.

Chapter 3

The 1-Way Classification

In Chapter 2 we reviewed several methods for variance components estimation and methods for statistical analysis with missing data. This chapter deals with variance components estimation with missing data from one-way model. Section 2 covers MLE for complete data. A result of an analytic expression of the inverse of a matrix will help us to avoid computing the inverse of a matrix at each iteration when we use a numerical procedure to find the MLE. A model-based method for estimation of variance components with missing data which we developed is described in section 3, and some examples are in section 4. Since the EM algorithm is a very general iterative algorithm for ML estimation in missing-data problems, we especially discuss EM algorithm for our case in section 5.

3.1 The Model

The one-way classification model is defined as

$$y_{ij} = \mu + \alpha_i + e_{ij}, \begin{cases} i = 1, 2, \dots, A \\ j = 1, 2, \dots, n_i \end{cases}$$

where μ is a general mean, the unobservable random variables α_i and e_{ij} have independent $N(0, \sigma_1^2)$ and $N(0, \sigma_0^2)$ distributions, respectively. It follows that $(y_{11}, \dots, y_{An_A})$

are jointly normally distributed with mean $\mu = (\mu, \dots, \mu)$ and

$$Var(Y) = V = \sigma_1^2 Z_1 Z_1' + \sigma_0^2 I$$

where $Z_1 Z_1'$ is a $\sum_{i=1}^A n_i \times \sum_{i=1}^A n_i$ matrix

$$Z_1 Z_1' = \begin{pmatrix} J_1 & 0 & \dots & 0 \\ 0 & J_2 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & J_p \end{pmatrix},$$

and J_i denotes an $n_i \times n_i$ matrix consisting of $1's$.

The unknown parameters are μ, σ_0^2, σ_1^2, the last two of which are the variance components.

3.2 MLE for Complete-Data

According to (2.1), we need to evaluate the determinant and inverse of V to compute $\log L(\mu, \sigma_0^2, \sigma_1^2; Y)$. If Y_i denote the vector of observations of those from ith group, Y_i and $Y_{i'}$ are independent for any $i \neq i'$. We use the notation Σ_i to denote the variance of Y_i, then $\det V = \prod_{i=1}^p \det \Sigma_i$, and

$$V^{-1} = \begin{pmatrix} \Sigma_1^{-1} & 0 & \dots & 0 \\ 0 & \Sigma_2^{-1} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \Sigma_p^{-1} \end{pmatrix}.$$

where $\Sigma_i = \sigma_0^2 I_i + \sigma_1^2 J_i$.

The following proposition enables us to compute $\det V$ and V^{-1} analytically (Rao and Kleffe, 1988).

Proposition 1 *Let* $\Sigma_i = \sigma_0^2 I_i + \tau_1^2 J_i$, *then*

(1) $\Sigma_i^{-1} = \frac{1}{\sigma_0^2} I_i - \frac{\sigma_1^2}{\sigma_0^4 + n_i \sigma_0^2 \sigma_1^2} J_i$;

(2) $\det(\Sigma_i) = (\sigma_0^2)^{n_i - 1}(\sigma_0^2 + n_i \sigma_1^2)$.

Therefore the log-likelihood function for one-way model becomes

$$\begin{aligned}
\log L &= C - \frac{1}{2}\sum_{i=1}^{A} \log[(\sigma_0^2)^{n_i-1}(\sigma_0^2 + n_i\sigma_1^2)] \\
&\quad - \frac{1}{2}\sum_{i=1}^{A}(Y_i - \mu_\mathbf{i})'(\frac{\mathbf{1}}{\sigma_\mathbf{0}^\mathbf{2}}\mathbf{I_i} - \frac{\sigma_\mathbf{1}^\mathbf{2}}{\sigma_\mathbf{0}^\mathbf{4} + \mathbf{n_i}\sigma_\mathbf{0}^\mathbf{2}\sigma_\mathbf{1}^\mathbf{2}}\mathbf{J_i})(\mathbf{Y_i} - \mu_\mathbf{i}) \\
&= C - \frac{1}{2}\sum_{i=1}^{A} \log[(\sigma_0^2)^{n_i-1}(\sigma_0^2 + n_i\sigma_1^2)] \\
&\quad - \frac{1}{2\sigma_0^2}\sum_{i=1}^{A}\sum_{j=1}^{n_i}(y_{ij} - \mu)^2 + \frac{1}{2}\sum_{i=1}^{A}\frac{\sigma_1^2}{\sigma_0^4 + n_i\sigma_0^2\sigma_1^2}[\sum_{j=1}^{n_i}(y_{ij} - \mu)]^2. \qquad (3.1)
\end{aligned}$$

The derivatives of the log-likelihood with respect to σ_0^2, σ_1^2, and μ, respectively, yield the following equations for the MLE's.

$$\begin{aligned}
\frac{\partial \log L}{\partial \sigma_0^2} &= -\frac{1}{2}\sum_{i=1}^{A}\frac{n_i\sigma_0^2 + n_i(n_i - 1)\sigma_1^2}{[(\sigma_0^2)(\sigma_0^2 + n_i\sigma_1^2)]} \\
&\quad + \frac{1}{2\sigma_0^4}\sum_{i=1}^{A}\sum_{j=1}^{n_i}(y_{ij} - \mu)^2 - \frac{1}{2}\sum_{i=1}^{A}\frac{\sigma_1^2(2\sigma_0^2 + n_i\sigma_1^2)}{(\sigma_0^4 + n_i\sigma_0^2\sigma_1^2)^2}[\sum_{j=1}^{n_i}(y_{ij} - \mu)]^2 \\
&= 0,
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \log L}{\partial \sigma_1^2} &= -\frac{1}{2}\sum_{i=1}^{A}\frac{n_i}{\sigma_0^2 + n_i\sigma_1^2} + \frac{1}{2}\sum_{i=1}^{A}\frac{\sigma_0^4}{(\sigma_0^4 + n_i\sigma_0^2\sigma_1^2)^2}[\sum_{j=1}^{n_i}(y_{ij} - \mu)]^2 \\
&= 0,
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial \log L}{\partial \mu} &= \frac{1}{\sigma_0^2}\sum_{i=1}^{A}\sum_{j=1}^{n_i}(y_{ij} - \mu) - \sum_{i=1}^{A}\frac{\sigma_1^2}{\sigma_0^4 + n_i\sigma_0^2\sigma_1^2}[\sum_{j-1}^{n_i}(y_{ij} - \mu)] \\
&= 0.
\end{aligned}$$

For balanced data ($n_1 = n_2 = \ldots = n_A$) , the maximum likelihood estimates of variance components are $\hat{\sigma_0^2} = MSE$ and $\hat{\sigma_1^2} = [SSTr/A - MSE]/n$, where

$$MSE = \frac{\sum_{i=1}^{A}\sum_{j=1}^{n}(y_{ij} - \bar{y}_i)^2}{A(n-1)},$$

and

$$SSTr = n\sum_{i=1}^{A}(y_i - \bar{y})^2.$$

For unbalanced data, the maximizing equations do not yield a explicit solutions and the MLE must be obtained by nonlinear optimization.

3.3 MLE for Incomplete Data

If

$$P(y_{ij}observed \mid y_{ij}) = \begin{cases} 1 & \text{if } y_{ij} > c \\ 0 & \text{otherwise,} \end{cases}$$

the mechanism leading to missing data here is called censoring, with observed values censored from below, or left censored, at c. This missing data mechanism is nonignorable because the probability that y_{ij} is observed depends on the value of y_{ij} (Little and Rubin, 1987).

If c is known, then we have the partial information about the random variable of interest. We know the distribution of the missing data and we also know that the missing value is less than c. The analysis of data needs to take this information into account to avoid biased results.

Suppose we have a one-way model in which the factor has A classes with n_i member of each class

$$\begin{array}{cccc} y_{11} & y_{12} & \cdots & y_{1n_1} \\ y_{21} & y_{22} & \cdots & y_{2n_2} \\ \vdots & \vdots & \ldots & \vdots \\ y_{A1} & y_{A2} & \cdots & y_{An_A}, \end{array}$$

let A_{obs} denote the number of observed classes, and m_i denote uncensored observations in class i. If we define $L(y_{i1}, \ldots, y_{im_i}, y_{im_i+1} < c, \ldots, y_{in_i} < c)$ as

$$\begin{aligned}
&L(y_{i1}, \ldots, y_{im_i}, y_{im_i+1} < c, \ldots, y_{in_i} < c) = \\
&\quad \lim_{\Delta \to 0} P(y_{i1} < Y_{i1} \leq y_{i1} + \Delta, \ldots, y_{im_i} < Y_{im_i} \leq y_{im_i} + \Delta, \ldots, \\
&\quad Y_{im_i+1} < c, \ldots, Y_{in_i} < c)
\end{aligned}$$

The overall likelihood function of $Y = (y_{11}, \ldots, y_{An_A})$ can be written as

$$\begin{aligned}
L \;=\; & L(y_{11}, \ldots, y_{1m_1}, y_{1m_1+1} < c, \ldots, y_{1n_1} < c; \\
& \ldots; \\
& y_{A_{obs}1}, \ldots, y_{A_{obs}m_{A_{obs}}}, y_{A_{obs}m_{A_{obs}}+1} < c, \ldots, y_{A_{obs}n_{A_{obs}}} < c; \\
& y_{A_{obs}+11} < c, \ldots, y_{A_{obs}+1n_{A_{obs}+1}} < c, y_{A1} < c, \ldots y_{An_A} < c)
\end{aligned}$$

where c is a known constant and L represents likelihood.

Let Y_i denote the vector of observations of those from ith class. Y_i and $Y_{i'}$ are independent for any $i \neq i'$. The likelihood of $Y = (Y_1, \ldots, Y_A)$ becomes

$$\begin{aligned}
L(Y) &= \prod_{i=1}^{A} L(Y_i) \\
&= \prod_{i=1}^{A_{obs}} L(y_{i1}, \ldots, y_{im_i}, y_{im_i+1} < c, \ldots, y_{in_i} < c) \\
&\quad \prod_{i=A_{obs}+1}^{A} L(y_{i1} < c, \ldots, y_{in_i} < c).
\end{aligned}$$

The log-likelihood is then

$$\begin{aligned}
\log L(\mu, V; Y) &= \sum_{i=1}^{A} \log L(Y_i) \\
&= \sum_{i=1}^{A_{obs}} \log L(y_{i1}, \ldots, y_{im_i}, y_{im_i+1} < c, \ldots, y_{in_i} < c)
\end{aligned}$$

$$+\sum_{i=A_{obs}+1}^{A} \log L(y_{i1} < c, \ldots, y_{in_i} < c). \tag{3.2}$$

In theory, σ_0^2, σ_1^2 and μ can be estimated by maximizing function (3.2). But

$$\lim_{\Delta \to 0} P(y_{i,obs.} < Y_{i,obs.} \leq y_{i,obs.} + \Delta, Y_{i,mis.} < c) = \int_{-\infty}^{c}, \ldots, \int_{-\infty}^{c} f(y_{i,obs.}, y_{i,mis.}) dy_{i,mis.}$$

is difficult to compute because of the dimension of $Y_{i,mis.}$ For the example in Chapter 1, the dimension of $y_{7,mis.}$ is 174.

Since each Y_i is a multivariate normal with mean μ and variance-covariance matrix

$$V_i = \sigma_0^2 I_i + \sigma_1^2 J_i,$$

the following proposition can be used to transform the $L(y_{i,obs.}, y_{i,mis.} < c)$ into a computable function.

Proposition 2 *If*

$$\begin{pmatrix} y_{i1} \\ y_{i2} \\ \vdots \\ y_{in_i} \end{pmatrix} \sim N(\mu, \sigma_0^2 I_i + \sigma_1^2 J_i),$$

then

$$L(y_{i1}, y_{i2}, \ldots, y_{im_i}, y_{im_i+1} < c, \ldots, y_{in_i} < c) \propto \int_{-\infty}^{\infty} \sigma_0^{-m_i} \Phi(w)^{n_i - m_i} \exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij} - \mu - \sigma_1 z_0)^2}{\sigma_0^2} - z_0^2]\} dz_0$$

where $w = \frac{c-\mu-\sigma_1 z_0}{\sigma_0}$.

For the proof see Chapter 5.

Applying the Proposition 2 to function (3.2) gives

$$\log L(\mu, V; Y) = \sum_{i=1}^{A} \log L(Y_i)$$

$$= \sum_{i=1}^{A_{obs}} \log \int_{-\infty}^{\infty} \sigma_0^{-m_i} \Phi(w)^{n_i - m_i} \exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij} - \mu - \sigma_1 y_0)^2}{\sigma_0^2} + z_0^2]\} dz_0$$

$$+ \sum_{i=A_{obs}+1}^{A} \log \int_{-\infty}^{\infty} \Phi(w)^{n_i} \exp(-\frac{z_0^2}{2}) dz_0 \qquad (3.3)$$

where $\Phi(w) = \int_{-\infty}^{w} \phi(x)dx$ and $w = \frac{c-\mu-\sigma_1 z_0}{\sigma_0}$.

It is important to note that the likelihood (3.3) now is one dimension integration which can be evaluated by Gaussian quadrature method. The σ_0^2, σ_1^2, and μ can be estimated by maximizing the above function in a given parameter space. To maximum the function, we can use a global search or an optimization routine.

The existence of a unique ML estimate for likelihood (3.3) is proved in chapter 5.

3.4 Examples

In this section, we apply the procedure which we developed above to two examples. The first example involves a one-way data set with and without missing data. The second example is based on a group of simulated data in which we have known parameters.

Example 1. A data set in Ott (p355) that involves a study of starch content of tomato plants grown in three different nutrients has been considered. The original data set is one with complete data as in table 3.1

The MLE with the complete data are

$$\hat{\sigma_0^2} = 8.289, \hat{\sigma_1^2} = 24.365, \hat{\mu} = 11.89$$

(both by SAS (PROC VARCOMP) and our own program).

Table 3.1: *Complete Ott's data*

A	22	20	21	18	16	14
B	12	14	15	10	9	6
C	7	9	7	6	5	3

Then four observations (< 6.99) were deleted as if they were missing where the missing pattern is

$$P(y_{ij} observed \mid y_{ij}) = \begin{cases} 1 & \text{if } y_{ij} > 6.99 \\ 0 & \text{otherwise.} \end{cases}$$

The incomplete data are in table 3.2

Using 14 data. estimates by our method (MLE_{mis}), and MLE are listed and compared with MLE with complete data as in table 3.3

As can be seen our method is more accurate in that it gives results very close to those of the MLE for the complete data.

Example 2. In this example, 10 groups of simulation data were randomly generated using $\sigma_0^2 = 1.0$, $\sigma_1^2 = 1.0$, and $\mu = 0.0$. The complete data are assumed to have five classes with each class having eight observations. In each simulation, the data whic' are less then -0.5 were removed from data set as if they were missing data (there is about 35 % of the data missing). The estimates and their mean squared errors (MSE) are given in the table 3.4

As we know, that MLE and ANOVA will subject to bias when observed values

Table 3.2: *Incomplete Ott's data*

A	22	20	21	18	16	14
B	12	14	15	10	9	
C	7	9	7			

Table 3.3: *Estimates by two methods compared with MLE with complete data*

	σ_0^2	σ_1^2	μ
MLE(18 data)	8.289	24.365	11.89
MLE(14 data)	6.92	18.13	12.83
MLE_{mis}(14 data)	8.93	24.17	11.91

	MLE			MLE_{mis}		
	$\hat{\sigma}_0^2$	$\hat{\sigma}_1^2$	$\hat{\mu}$	$\hat{\sigma}_0^2$	$\hat{\sigma}_1^2$	$\hat{\mu}$
1	0.5016	0.0992	0.6007	1.2098	0.2613	0.0100
2	0.3598	0.1819	0.8531	0.7349	0.0100	0.6382
3	0.9496	0.3217	1.3234	1.4501	0.8545	0.6386
4	0.5378	0.1786	0.6719	1.1515	0.1603	0.1500
5	0.2554	0.0114	0.3475	0.6511	0.0271	0.0100
6	0.9811	0.3464	1.4959	1.3114	0.9202	1.1428
7	0.6834	0.3382	0.9286	0.7068	0.8986	0.5780
8	0.4563	0.3419	0.6280	1.0176	0.9084	0.0100
9	0.5229	0.0488	0.4806	0.7090	0.1268	0.0100
10	0.7239	0.0806	0.3673	0.9127	0.2117	0.0100
MSE	0.2129	0.6641	0.7272	0.0738	0.1608	0.2181

Table 3.4: Comparison of MLE and MLE_{mis} for 10 Simulations

censored from below (or left censored). The reasons that I compared our method with MLE and ANOVA are

- no method is available to this variance components estimation with censoring data;
- biologists use MLE or ANOVA to estimate the variance components when only the incomplete data can be obtained.

Compared with MLE, MSE are improved by using the present method, reducing the MSE by 66% for σ_0^2, 31% for σ_1^2, and 66% for μ respectively. Significant improvements are obtained through our method.

3.5 The EM Algorithm

An iterative algorithm for calculating ML estimates in missing-data problem is EM algorithm. Its name stands for Expectation-Maximization, and it is so named because it alternates between E step and M step (Little and Rubin, 1987).

The E step finds the conditional expectation

$$Q(\theta \mid \theta^t) = \int l(\theta \mid Y) f(Y_{mis} \mid y_{obs}, \theta = \theta^t) dY_{mis}$$

where $l(\theta \mid Y)$ is the function of Y_{mis} and Y_{obs} appearing in the complete-data log-likelihood.

The M step performs maximum likelihood estimation of parameters just as if there were no missing data. Thus the M step of EM uses the identical computational methods as ML estimation from likelihood $l(\theta \mid Y)$. θ^{t+1} is determined by

$$Q(\theta^{t+1} \mid \theta^t) = max_\theta Q(\theta \mid \theta^t)$$

We now show how to use EM steps to estimate variance components with missing data. To do so, consider the model, observed data and missing data in section 3.

Whether we can apply the EM algorithm to get variance components estimation with missing data is based on our ability to calculate the conditional expectation. For this we need the joint distribution of (Y_{obs}, Y_{mis}) and the condition distribution of Y_{mis} given Y_{obs}. From section 2, we obtain the joint density

$$\begin{aligned}
\log L &= C - \frac{1}{2}\sum_{i=1}^{A} \log[(\sigma_0^2)^{n_i-1}(\sigma_0^2 + n_i\sigma_1^2)] \\
&\quad - \frac{1}{2\sigma_0^2}\sum_{i=1}^{A}((Y_{iobs} - \mu_{iobs})'(Y_{iobs} - \mu_{iobs}) + ((Y_{imis} - \mu_{imis})'(Y_{imis} - \mu_{imis}) \\
&\quad + \frac{\sigma_1^2}{2(\sigma_0^4 + n_i\sigma_0^2\sigma_1^2)}\sum_{i=1}^{A}(1'_{m_i \times 1}(Y_{iobs} - \mu_{iobs}) + 1'_{(n_i - m_i)\times 1}(Y_{imis} - \mu_{imis}))^2.
\end{aligned}$$

From the following properties which Searle (1992) and many other texts have shown, we can get conditional distribution and some conditional expectation straight forward.

On writing

$$\begin{pmatrix} Y_{iobs} \\ Y_{imis} \end{pmatrix} \sim N\left(\begin{pmatrix} \mu_{iobs} \\ \mu_{imis} \end{pmatrix}, \begin{pmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{pmatrix}\right),$$

- the marginal distribution of Y_{imis} is

$$Y_{imis} \sim N(\mu_{imis}, V_{22});$$

- the conditional distribution of Y_{imis} given Y_{iobs} is

$$Y_{imis} \mid Y_{iobs} \sim N(\mu_{imis} + V_{12}V_{22}^{-1}(Y_{iobs} - \mu_{iobs}), V_{22} - V_{21}V_{11}^{-1}V_{12});$$

- and

$$E(Y'_{imis}Y_{imis}) = tr(V_{22}) + \mu'_{imis}\mu_{imis}.$$

Therefore the E step here calculates

$$\begin{aligned}
&E(\log L \mid Y_{obs}) = \\
&\quad C - \frac{1}{2}\sum_{i=1}^{A} \log[(\sigma_0^2)^{n_i-1}(\sigma_0^2 + n_i\sigma_1^2)] \\
&\quad -\frac{1}{2\sigma_0^2}\sum_{i=1}^{A}((Y_{iobs} - \mu_{iobs})'(Y_{iobs} - \mu_{iobs}) + (tr(V_{22} + \mu'_{imis}\mu_{imis})) \\
&\quad +\frac{\sigma_1^2}{2(\sigma_0^4 + n_i\sigma_0^2\sigma_1^2)}\sum_{i=1}^{A} E((1'_{m_i\times 1}(Y_{iobs} - \mu_{iobs}) \\
&\quad +1'_{(n_i-m_i)\times 1}(Y_{imis} - \mu_{imis}))^2 \mid Y_{iobs}),
\end{aligned}$$

and the M step calculates

$$max_{(\sigma_0^2,\sigma_1^2,\mu)} E(\log L \mid Y_{obs}).$$

After the details of computing have been outlined above, we can see that the last term of $E(\log L \mid Y_{obs})$

$$E((1'_{m_i\times 1}(Y_{iobs} - \mu_{iobs}) + 1'_{(n_i-m_i)\times 1}(Y_{imis} - \mu_{imis}))^2 \mid Y_{iobs})$$

can be terribly complicated. Let us look one term of above condition expectation

$$E(1'_{(n_i-m_i)\times 1}(Y_{imis}-\mu_{imis}))^2 \mid Y_{iobs}) = \int (\sum_{j=m_i+1}^{n_i}(y_{ij}-\mu_{ij}))^2 f(y_{imis} \mid y_{iobs}, \theta = \theta^t) dy_{i,mis}$$

which involves multidimensional integration. The computation is difficult when dimension of $Y_{i,mis}$ is large. We can see it is hard to apply EM algorithm to estimate variance components with incomplete Y even with one-way model.

Chapter 4

The 2-way Nested Classification

Detailed results of variance components estimation with missing data from 2 way nested model are provided in this chapter. In section 2, we derive a log likelihood without the inverse of a matrix. That enables us to evaluate the log likelihood more accurately and efficiently, especially for large data sets. Section 3 is the central section of this chapter, presenting the new method. Examples are given in section 4.

4.1 The Model

Two-way nested designs are used widely in applied breeding (Henderson, 1984). Consider the following breeding experiment

Sire	i	1			...	S		
Dam	j	1	...	D_1	...	1	...	D_S
OffSpring	k	1	1	1	...	1	1	1
		.	.	.	...	.	.	.
		n_{11}	.	n_{1D_1}	...	n_{S1}	.	n_{SD}

There are S sires, D_i dams were mated to the ith sire, and n_{ij} offspring resulted from each $i-j$ mating. If y_{ijk} is a characteristic (such are length, weight, *etc.*)

measured on the kth offspring of the ijth mating, the structure of y_{ijk} is

$$y_{ijk} = \mu + \alpha_i + \beta_{j(i)} + \epsilon_{k(ij)}, \begin{cases} i = 1, 2, \ldots, S \\ j = 1, 2, \ldots, D_i \\ k = 1, 2, \ldots, n_{ij} \end{cases}$$

where μ is an unknown constant, α_i, $\beta_{j(i)}$, and $\epsilon_{k(ij)}$ are mutually independent $N(0, \sigma_1^2)$, $N(0, \sigma_2^2)$, $N(0, \sigma_0^2)$ respectively. α_i is the contribution due to the ith sire, $\beta_{j(i)}$ is due to the jth dam mated to the ith sire, and $\epsilon_{k(ij)}$ is the effect due to the kth offspring of the ijth sire-dam mating. The quantities σ_1^2, σ_2^2 and σ_0^2 are called variance components. The heritability $h^2 = \frac{4\sigma_1^2}{\sigma_1^2+\sigma_2^2+\sigma_0^2}$ is a function of variance components (Falconer, 1981).

4.2 The Complete-Data

4.2.1 Balanced data

When $D_i = D$ for all i, and each ij cell contains the same number of offspring ($n_{ij} = n$ for all i and j), the data shall be described as balanced data. Estimating variance components from balanced data is, generally speaking, much easier than from unbalanced data.

Follow the same notation from previous context, so the linear model is given by

$$Y = \mu + Z_1\gamma_1 + Z_2\gamma_2 + \epsilon,$$

where μ is an $SDn \times 1$ vector of unknown constants;
Z_1 is an $SDn \times S$ design matrix of zeros and ones;
γ_1 is an $S \times 1$ vector of independent variables from $N(0, \sigma_1^2)$;
Z_2 is an $SDn \times SD$ design matrix of zeros and ones;
γ_2 is an $SD \times 1$ vector of independent variables from $N(0, \sigma_2^2)$;
ϵ is an $SDn \times 1$ vector of independent variables from $N(0, \sigma_0^2)$.

The random vectors $\gamma_1, \gamma_2, \epsilon$ are mutually independent and

$$V = Var(Y) = Z_1 Z_1^T \sigma_1^2 + Z_2 Z_2^T \sigma_2^2 + I\sigma_0^2.$$

$Z_1 Z_1^T$ and $Z_2 Z_2^T$ are $SDn \times SDn$ block-diagonal matrices:

$$Z_1 Z_1^T = \begin{pmatrix} J & 0 & \dots & 0 \\ 0 & J & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & J \end{pmatrix}$$

and

$$Z_2 Z_2^T = \begin{pmatrix} E & 0 & \dots & 0 \\ 0 & E & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & E \end{pmatrix},$$

where J and E denote respectively $Dn \times Dn$ and $n \times n$ matrices consisting of 1's. Therefore we can write

$$V = \begin{pmatrix} \Sigma & 0 & \dots & 0 \\ 0 & \Sigma & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \Sigma \end{pmatrix},$$

where $\Sigma = \sigma_0^2 I + \sigma_1^2 J + \sigma_2^2 K$ and

$$K = \begin{pmatrix} E & 0 & \dots & 0 \\ 0 & E & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & E \end{pmatrix}.$$

Thus the log-likelihood function of Y is given by

$$\log L(\mu, V; Y) = c - \frac{1}{2}\sum_{i=1}^{S}[\log(\det(\Sigma)) + (Y_i - \mu_i)'\Sigma^{-1}(Y_i - \mu_i)]. \tag{4.1}$$

In order to proceed with finding the maximum likelihood, the following proposition on the determinant and the inverse of Σ is needed. When we use an iterative numerical procedure to find the maximum of $\log L$, often the computational effort is dominated by the cost of evaluating $\log L$. Using the analytical expression below for the inverse of V, we do not need to compute the inverse of a matrix to evaluate $\log L$ at each iteration (Rao and Kleffe, 1988). This has advantages both in accuracy and efficiency.

Proposition 3 *Let* $\Sigma_{Dn\times Dn} = \sigma_0^2 I_{Dn\times Dn} + \sigma_1^2 J_{Dn\times Dn} + \sigma_2^2 K_{Dn\times Dn}$, *then*

(1) $\Sigma^{-1}_{Dn\times Dn} = \frac{1}{\sigma_0^2} I_{Dn\times Dn} - \frac{\sigma_1^2}{(\sigma_0^2+n\sigma_2^2)(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2)} J_{Dn\times Dn} - \frac{\sigma_2^2}{\sigma_0^2(\sigma_0^2+n\sigma_2^2)} K_{Dn\times Dn}$

(2) $\det(\Sigma) = (\sigma_0^2)^{Dn-D}(\sigma_0^2+n\sigma_2^2)^{D-1}(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2)$

where $J_{Dn\times Dn} = 1_{Dn\times 1}1^T_{Dn\times 1}$ *and*

$$K_{Dn\times Dn} = \begin{pmatrix} J_{n\times n} & 0 & \dots & 0 \\ 0 & J_{n\times n} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & J_{n\times n} \end{pmatrix}.$$

PROOF. (1):

Let

$$\begin{aligned} A_{Dn\times Dn} &= \Sigma_{Dn\times Dn} - \sigma_1^2 J_{Dn\times Dn} \\ &= \sigma_0^2 I_{Dn\times Dn} + \sigma_2^2 K_{Dn\times Dn}, \end{aligned}$$

then

$$\begin{aligned} \Sigma &= \Sigma - \sigma_1^2 J_{Dn\times Dn} + \sigma_1^2 J_{Dn\times Dn} \\ &= (\Sigma - \sigma_1^2 J_{Dn\times Dn})[I + (\Sigma - \sigma_1^2 J_{Dn\times Dn})^{-1}\sigma_1^2 J_{Dn\times Dn}] \\ &= A_{Dn\times Dn}[I_{Dn\times Dn} + A^{-1}\sigma_1^2 J_{Dn\times Dn}]. \end{aligned}$$

We can write

$$\Sigma^{-1}_{Dn\times Dn} = [I_{Dn\times Dn} + \sigma_1^2 A_{-1} J_{Dn\times Dn}]^{-1} A^{-1}_{Dn\times Dn}. \qquad (1.2)$$

Since

$$A_{Dn\times Dn} = \begin{pmatrix} \sigma_0^2 I_{n\times n} + \sigma_2^2 J_{n\times n} & 0 & \dots & 0 \\ 0 & \sigma_0^2 I_{n\times n} + \sigma_2^2 J_{n\times n} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \sigma_0^2 I_{n\times n} + \sigma_2^2 J_{n\times n} \end{pmatrix},$$

we have

$$A^{-1} = \begin{pmatrix} \frac{1}{\sigma_0^2} I_{n\times n} + \frac{\sigma_2^2 J_{n\times n}}{\sigma_0^4 + n\sigma_2^2\sigma_0^2} & 0 & \dots & 0 \\ 0 & \frac{1}{\sigma_0^2} I_{n\times n} + \frac{\sigma_2^2 J_{n\times n}}{\sigma_0^4 + n\sigma_2^2\sigma_0^2} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \frac{1}{\sigma_0^2} I_{n\times n} + \frac{\sigma_2^2 J_{n\times n}}{\sigma_0^4 + n\sigma_2^2\sigma_0^2} \end{pmatrix}$$

or

$$A^{-1} = \frac{1}{\sigma_0^2} I_{Dn\times Dn} - \frac{\sigma_2^2}{\sigma_0^4 + n\sigma_2^2\sigma_0^2} K_{Dn\times Dn}$$

by Proposition 1.

Also

$$\begin{aligned} \sigma_1^2 A^{-1} J_{Dn\times Dn} &= \frac{\sigma_1^2}{\sigma_0^2} J_{Dn\times Dn} - \frac{n\sigma_1^2\sigma_2^2}{\sigma_0^2(\sigma_0^2 + n\sigma_2^2)} J_{Dn\times Dn} \\ &= \frac{\sigma_1^2}{\sigma_0^2 + n\sigma_2^2} J_{Dn\times Dn}. \end{aligned}$$

It is easily seen that

$$\begin{aligned} \Sigma^{-1} &= [I_{Dn\times Dn} + \sigma_1^2 A^{-1} J_{Dn\times Dn}]^{-1} A^{-1} \\ &= [I_{Dn\times Dn} + \frac{\sigma_1^2}{\sigma_0^2 + n\sigma_2^2} J]^{-1} [\frac{1}{\sigma_0^2} I_{Dn\times Dn} - \frac{\sigma_2^2}{\sigma_0^4 + n\sigma_2^2\sigma_0^2} K_{Dn\times Dn}] \\ &= [I_{Dn\times Dn} - \frac{\sigma_1^2}{\sigma_0^2 + n\sigma_2^2 + Dn\sigma_1^2} J][\frac{1}{\sigma_0^2} I_{Dn\times Dn} - \frac{\sigma_2^2}{\sigma_0^4 + n\sigma_2^2\sigma_0^2} K_{Dn\times Dn}] \end{aligned}$$

$$= \frac{1}{\sigma_0^2} I_{Dn\times Dn} - \frac{\sigma_1^2}{(\sigma_0^2+n\sigma_2^2)(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2)} J_{Dn\times Dn}$$
$$-\frac{\sigma_2^2}{\sigma_0^2(\sigma_0^2+n\sigma_2^2)} K_{Dn\times Dn}.$$

□

PROOF. (2):

From equation 4.2, we have

$$\Sigma_{Dn\times Dn} = A_{Dn\times Dn}[I_{Dn\times Dn} + \sigma_1^2 A^{-1}_{Dn\times Dn} J_{Dn\times Dn}].$$

It follows that

$$\det(\Sigma) = \det(A)\det(I_{Dn\times Dn} + \sigma_1^2 A^{-1} J_{Dn\times Dn}).$$

Note that

$$\begin{aligned} \det(A) &= [\det(\sigma_0^2 I_{n\times n} + \sigma_2^2 J_{n\times n})]^D \\ &= [(\sigma_0^2)^{n-1}(\sigma_0^2+n\sigma_2^2)]^D \end{aligned}$$

and

$$\begin{aligned} \det(I + \sigma_1^2 A^{-1} J_{Dn\times Dn}) &= \det(I + \frac{\sigma_1^2}{\sigma_0^2+n\sigma_2^2} J_{n\times n}) \\ &= (1 + \frac{Dn\sigma_1^2}{\sigma_0^2+n\sigma_2^2}) \end{aligned}$$

by Proposition 1, $\det(\Sigma)$ becomes

$$(\sigma_0^2)^{Dn-D}(\sigma_0^2+n\sigma_2^2)^{D-1}(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2).$$

□

Using Proposition 3 to function 4.1, we obtain

$$\begin{aligned} \log L &= C - \frac{S}{2}\log[(\sigma_0^2)^{Dn-D}(\sigma_0^2+n\sigma_2^2)^{D-1}(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2)] \\ &\quad - \frac{1}{2\sigma_0^2}\sum_{i=1}^{S}\sum_{j=1}^{D}\sum_{k=1}^{n}(y_{ijk}-\mu)^2 \\ &\quad + \frac{\sigma_1^2}{2(\sigma_0^2+n\sigma_2^2)(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2)}\sum_{i=1}^{S}\{\sum_{j=1}^{D}\sum_{k=1}^{n}(y_{ijk}-\mu)\}^2 \\ &\quad + \frac{\sigma_2^2}{2\sigma_0^2(\sigma_0^2+n\sigma_2^2)}\sum_{i=1}^{S}\sum_{j=1}^{D}[\sum_{k=1}^{n}(y_{ijk}-\mu)]^2. \end{aligned} \tag{4.3}$$

The differentiation of (4.3) with regard to $\sigma_0^2, \sigma_1^2, \sigma_2^2$ and μ yields the following equations for the MLE's.

$$
\begin{aligned}
\frac{\partial \log L}{\partial \sigma_0^2} &= \\
& \frac{-S}{2(\sigma_0^2)^{Dn-D}(\sigma_0^2+n\sigma_2^2)^{D-1}(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2)} \\
& [(Dn-D)(\sigma_0^2)^{Dn-D-1}(\sigma_0^2+n\sigma_2^2)^{D-1}(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2) \\
& +(\sigma_0^2)^{Dn-D}(D-1)(\sigma_0^2+n\sigma_2^2)^{D-2}(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2)+(\sigma_0^2)^{Dn-D}(\sigma_0^2+n\sigma_2^2)^{D-1}] \\
& +\frac{1}{2\sigma_0^4}\sum_{i=1}^{S}\sum_{j=1}^{D}\sum_{k=1}^{n}(y_{ijk}-\mu)^2-\frac{\sigma_1^2(2\sigma_0^2+2n\sigma_2^2+Dn\sigma_1^2)}{2[(\sigma_0^2+n\sigma_2^2)(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2]^2} \\
& \times\sum_{i=1}^{S}[\sum_{j=1}^{D}\sum_{k=1}^{n}(y_{ijk}-\mu)]^2-\frac{\sigma_2^2(2\sigma_0^2+n\sigma_2^2)}{2[(\sigma_0^4+n\sigma_0^2\sigma_2^2)]^2}\sum_{i=1}^{S}\sum_{j=1}^{D}[\sum_{k=1}^{n}(y_{ijk}-\mu)]^2 \\
&= 0,
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \log L}{\partial \sigma_1^2} &= \frac{-SDn(\sigma_0^2)^{Dn-D}(\sigma_0^2+n\sigma_2^2)^{D-1}}{2(\sigma_0^2)^{Dn-D}(\sigma_0^2+n\sigma_2^2)^{D-1}(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2)} \\
&\quad +\frac{(\sigma_0^2+n\sigma_2^2)^2}{2[(\sigma_0^2+n\sigma_2^2)(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2]^2}\sum_{i=1}^{S}[\sum_{j=1}^{D}\sum_{k=1}^{n}(y_{ijk}-\mu)]^2 \\
&= 0,
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \log L}{\partial \sigma_2^2} &= \\
& -\frac{S[(\sigma_0^2)^{Dn-D}n(D-1)(\sigma_0^2+n\sigma_2^2)^{D-2}(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2)}{2(\sigma_0^2)^{Dn-D}(\sigma_0^2+n\sigma_2^2)^{D-1}(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2)} \\
& +\frac{n(\sigma_0^2)^{Dn-D}(\sigma_0^2+n\sigma_2^2)^{D-1}]}{2(\sigma_0^2)^{Dn-D}(\sigma_0^2+n\sigma_2^2)^{D-1}(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2)} \\
& -\frac{n\sigma_1^2(2\sigma_0^2+2n\sigma_2^2+Dn\sigma_1^2)}{2[(\sigma_0^2+n\sigma_2^2)(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2]^2}\sum_{i=1}^{S}[\sum_{j=1}^{D}\sum_{k=1}^{n}(y_{ijk}-\mu)]^2 \\
& -\frac{\sigma_0^4}{2[\sigma_0^4+n\sigma_0^2\sigma_2^2]^2}\sum_{i=1}^{S}\sum_{j=1}^{D}[\sum_{k=1}^{n}(y_{ijk}-\mu)]^2 \\
&= 0
\end{aligned}
$$

and

$$\begin{aligned}
\frac{\partial \log L}{\partial \mu} &= \frac{1}{\sigma_0^2}\sum_{i=1}^{S}\sum_{j=1}^{D}\sum_{k=1}^{n}(y_{ijk}-\mu) \\
&\quad - \frac{\sigma_1^2}{(\sigma_0^2+n\sigma_2^2)(\sigma_0^2+n\sigma_2^2+Dn\sigma_1^2)}\sum_{i=1}^{S}\sum_{j=1}^{D}\sum_{k=1}^{n}(y_{ijk}-\mu) \\
&\quad - \frac{\sigma_2^2}{\sigma_0^2(\sigma_0^2+n\sigma_2^2)}\sum_{i=1}^{S}\sum_{j=1}^{D}\sum_{k=1}^{n}(y_{ijk}-\mu) \\
&= 0.
\end{aligned}$$

Solutions to the above equations will give the ML estimates of the variance components.

4.2.2 Unbalanced data

Unbalanced data are those in which the numbers of observations in the subclasses of the model are not all the same, including cases where there are no observations in some subclasses. The estimation of variance components from unbalanced data is more complicated than from balanced data.

For unbalanced data, the variance of Y is

$$V = Var(Y) = Z_1Z_1^T\sigma_1^2 + Z_2Z_2^T\sigma_2^2 + I\sigma_0^2.$$

where $Z_1Z_1^T$ and $Z_2Z_2^T$ are $\sum_{i=1}^{S}\sum_{j=1}^{D_i} n_{ij} \times \sum_{i=1}^{S}\sum_{j=1}^{D_i} n_{ij}$ block-diagonal matrices:

$$Z_1Z_1^T = \begin{pmatrix} J_1 & 0 & \dots & 0 \\ 0 & J_2 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & J_S \end{pmatrix}$$

and

$$Z_2Z_2^T = \begin{pmatrix} E_{11} & 0 & \dots & 0 \\ 0 & E_{12} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & E_{1D_1} & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & E_{SD_S} \end{pmatrix}.$$

J_i and E_{ij} denote respectively $\sum_{j=1}^{D_i} n_{ij} \times \sum_{j=1}^{D_i} n_{ij}$ and $n_{ij} \times n_{ij}$ matrices consisting of 1′s. Therefore we can write

$$V = \begin{pmatrix} \Sigma_1 & 0 & \dots & 0 \\ 0 & \Sigma_2 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \Sigma_S \end{pmatrix},$$

where $\Sigma_i = \sigma_0^2 I_i + \sigma_1^2 J_i + \sigma_2^2 K_i$ and

$$K_i = \begin{pmatrix} E_{i1} & 0 & \dots & 0 \\ 0 & E_{i2} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & E_{iD_i} \end{pmatrix}.$$

The following are results about $\det(\Sigma_i)$ and Σ_i^{-1}. For the same reason as we mentioned in the previous section, an analytical expression for the inverse of V can avoid the heavy computational burden when we use numerical techniques to calculate the estimates (Rao and Kleffe, 1988).

Proposition 4 *Let* $\Sigma_i = \sigma_0^2 I_i + \sigma_1^2 J_i + \sigma_2^2 K_i$, *then*

(1) $\det(\Sigma_i) = (\sigma_0^2)^{(\sum_{j=1}^{D_i} n_{ij})-D_i} \prod_{j=1}^{D_i} (\sigma_0^2 + n_{ij}\sigma_2^2)[1 + \sigma_1^2 \sum_{j=1}^{D_i} \frac{n_{ij}}{\sigma_0^2 + n_{ij}\sigma_2^2}]$

(2) $\Sigma_i^{-1} = \frac{1}{\sigma_0^2} I_i - BJ - \frac{1}{1+\sigma_1^2 \sum_{j=1}^{D_i} \frac{n_{ij}}{\sigma_0^2 + n_{ij}\sigma_2^2}} * JK$

where

$$BJ = \begin{pmatrix} \frac{\sigma_2^2}{\sigma_0^2(\sigma_0^2+n_{i1}\sigma_2^2)} J_{n_{i1}\times n_{i1}} & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & \frac{\sigma_2^2}{\sigma_0^2(\sigma_0^2+n_{iD_i}\sigma_2^2)} J_{n_{iD_i}\times n_{iD_i}} \end{pmatrix},$$

and

$$JK = \begin{pmatrix} \frac{\sigma_1}{\sigma_0^2+n_{i1}\sigma_2^2} \\ \vdots \\ \frac{\sigma_1}{\sigma_0^2+n_{iD_i}\sigma_2^2} \end{pmatrix} * \begin{pmatrix} \frac{\sigma_1}{\sigma_0^2+n_{i1}\sigma_2^2}, \dots, \frac{\sigma_1}{\sigma_0^2+n_{iD_i}\sigma_2^2} \end{pmatrix}.$$

PROOF. (1):

Let $A_i = \Sigma_i - \sigma_1^2 J_i$, then

$$\Sigma_i = A_i[I + \sigma_1^2(A_i)^{-1}J_i].$$

This gives

$$\det \Sigma_i = \det A_i \det I + \sigma_1^2(A_i)^{-1}J_i.$$

Note that

$$\det A_i = (\sigma_0^2)^{\sum_{j=1}^{D_i} n_{ij} - D_i} \prod_{j=1}^{D_i}(\sigma_0^2 + n_{ij}\sigma_2^2)$$

by Proposition 1 and

$$\det I + \sigma_1^2(A_i)^{-1}J_i = 1 + \sigma_1^2 \sum j = 1^{D_i} \frac{n_{ij}}{\sigma_0^2 + n_{ij}\sigma_2^2},$$

thus

$$\det(\Sigma_i) = (\sigma_0^2)^{(\sum_{j=1}^{D_i} n_{ij}) - D_i} \prod_{j=1}^{D_i}(\sigma_0^2 + n_{ij}\sigma_2^2)[1 + \sigma_1^2 \sum_{j=1}^{D_i} \frac{n_{ij}}{\sigma_0^2 + n_{ij}\sigma_2^2}].$$

□

PROOF. (2):

Since

$$(A_i)^{-1} = \begin{pmatrix} \frac{1}{\sigma_0^2} I - \frac{\sigma_2^2}{\sigma_0^4 + n_{i1}\sigma_0^2\sigma_2^2} J_{n_{i1}\times n_{i1}} & 0 & \ldots & 0 \\ 0 & \Sigma_2^{-1} & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & . & \frac{1}{\sigma_0^2} I - \frac{\sigma_2^2}{\sigma_0^4 + n_{iD_i}\sigma_0^2\sigma_2^2} J_{n_{iD_i}\times n_{iD_i}} \end{pmatrix},$$

and

$$[I + \sigma_1^2 (A_i)^{-1} J_i]^{-1} = I - \frac{1}{1 + \sigma_1^2 \sum_{j=1}^{D_i} \frac{n_{ij}}{\sigma_0^2 + n_{ij}\sigma_2^2}} \begin{pmatrix} \frac{\sigma_1^2}{\sigma_0^2 + n_{i1}\sigma_2^2} \\ \vdots \\ \frac{\sigma_1^2}{\sigma_0^2 + n_{iD_i}\sigma_2^2} \end{pmatrix} 1^t,$$

we obtain

$$\Sigma_i^{-1} = \frac{1}{\sigma_0^2} I - BJ - \frac{1}{1 + \sigma_1^2 \sum_{j=1}^{D_i} \frac{n_{ij}}{\sigma_0^2 + n_{ij}\sigma_2^2}} * JK$$

Where

$$BJ = \begin{pmatrix} \frac{\sigma_2^2}{\sigma_0^2(\sigma_0^2 + n_{i1}\sigma_2^2)} J_{n_{i1}\times n_{i1}} & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \vdots & \vdots & .. & \vdots \\ 0 & 0 & \ldots & \frac{\sigma_2^2}{\sigma_0^2(\sigma_0^2 + n_{iD_i}\sigma_2^2)} J_{n_{iD_i}\times n_{iD_i}} \end{pmatrix},$$

and

$$JK = \begin{pmatrix} \frac{\sigma_1}{\sigma_0^2 + n_{i1}\sigma_2^2} \\ \vdots \\ \frac{\sigma_1}{\sigma_0^2 + n_{iD_i}\sigma_2^2} \end{pmatrix} * \left(\frac{\sigma_1}{\sigma_0^2 + n_{i1}\sigma_2^2}, \ldots, \frac{\sigma_1}{\sigma_0^2 + n_{iD_i}\sigma_2^2} \right).$$

□

The log likelihood function now becomes

$$\begin{aligned}\log L &= C - \frac{1}{2}\sum_{i=1}^{S}\{[(\sum_{j=1}^{D_i} n_{ij}) - D_i]\log\sigma_0^2 + \sum_{j=1}^{D_i}\log(\sigma_0^2 + n_{ij}\sigma_2^2) \\ &\quad + \log[1 + \sigma_1^2\sum_{j=1}^{D_i}\frac{n_{ij}}{\sigma_0^2 + n_{ij}\sigma_2^2}] + \frac{1}{\sigma_0^2}\sum_{j=1}^{D_i}\sum_{k=1}^{n_{ij}}(y_{ijk} - \mu)^2 \\ &\quad - \sum_{j=1}^{D_i}\frac{\sigma_2^2}{\sigma_0^2(\sigma_0^2 + n_{ij}\sigma_2^2)}[\sum_{k=1}^{n_{ij}}(y_{ijk} - \mu)]^2 \\ &\quad - \frac{\sigma_1^2}{1 + \sigma_1^2\sum_{j=1}^{D_i}\frac{n_{ij}}{\sigma_0^2 + n_{ij}\sigma_2^2}}[(\sum_{j=1} D_i\frac{1}{\sigma_0^2 + n_{ij}\sigma_2^2}(\sum_{k=1}^{n_{ij}}(y_{ijk} - \mu))]^2\} \qquad (4.4)\end{aligned}$$

The ML estimates of $\sigma_0^2, \sigma_1^2, \sigma_2^2$, and μ can be obtained by maximizing the above constrained nonlinear function.

There are many numerical procedures for the constrained nonlinear optimization problems, such as the steepest ascent algorithm and Newton-Raphson algorithm. There is no single iterative numerical algorithm for MLE that will be the best for every applications. Several computational algorithms are discussed by Harville (1977).

4.3 MLE for Incomplete Data

This section considers the estimation of variance components in the presence of censoring data.

Suppose y_{ijk} are observed for $i = 1, \ldots, S_{obs.}$; $j = 1, \ldots, D_{i,obs.}$; $k = 1, \ldots, m_{ij}$ and missing for $i = S_{obs.} + 1, \ldots, S$; $j = D_{i,obs.} + 1, \ldots, D_i$; $k = m_{ij} + 1, \ldots, n_{ij}$ in each $i - j$ mating. The missing-data mechanism is

$$P(y_{ijk} observed \mid y_{ijk}) = \begin{cases} 1 & \text{if } y_{ijk} > c \\ 0 & \text{otherwise.} \end{cases}$$

The missing data here is non-ignorable since the probability of response depends on the value of y_{ijk}. n_{ij} is the number of complete data in the ijth mating and m_{ij}

is the number of observed responses in the ijth mating, so that $n_{ij} - m_{ij}$ will be the number of missing observations in the ijth mating. To characterize the data, we define

$$\begin{aligned}
& L(y_{i11}, \ldots, y_{i1m_{i1}}, y_{i1m_{i1}+1} < c, \ldots, y_{i1n_{i1}} < c; \ldots; y_{iD_{i,obs}1}, \ldots, y_{iD_{i,obs}m_{iD_{i,obs}}}, \\
& \quad y_{iD_{i,obs}m_{iB_i}+1} < c, \ldots, y_{iD_{i,obs}n_{iD_{i,obs}}} < c; y_{iD_{i,obs}+11} < c, \ldots, \\
& \quad y_{iD_{i,obs}+1m_{iD_{i,obs}}} < c, \ldots, y_{iD_{i,obs}n_{iD_{i,obs}}} < c) \\
= & \lim_{\Delta \to 0} P(y_{i11} < Y_{i11} \leq y_{i11} + \Delta, \ldots, y_{i1m_{i1}} < Y_{i1m_{i1}} \leq y_{i1m_{i1}} + \Delta, \\
& \quad Y_{i1m_{i1}+1} < c, \ldots, Y_{i1n_{i1}} < c; \ldots; y_{iD_{i,obs}1} < Y_{iD_{i,obs}1} \leq y_{iD_{i,obs}1} + \Delta, \ldots, \\
& \quad y_{iD_{i,obs}m_{iD_{i,obs}}} < Y_{iD_{i,obs}m_{iD_{i,obs}}} \leq y_{iD_{i,obs}m_{iD_{i,obs}}} + \Delta; Y_{iD_{i,obs}m_{iB_i}+1} < c, \ldots; \\
& \quad Y_{iD_{i,obs}n_{iD_{i,obs}}} < c; Y_{iD_{i,obs}+11} < c, \ldots, Y_{iD_{i,obs}+1m_{iD_{i,obs}}} < c, \ldots, \\
& \quad Y_{iD_{i,obs}n_{iD_{i,obs}}} < c),
\end{aligned}$$

then the full likelihood function can be written as

$$\begin{aligned}
L(\mu, \sigma_0^2, \sigma_1^2, \sigma_2^2; Y) = & \prod_{i=1}^{S_{obs}} L(y_{i11}, \ldots, y_{i1m_{i1}}, y_{i1m_{i1}+1} < c, \ldots, y_{i1n_{i1}} < c; \\
& \qquad \ldots\ldots\ldots \\
& \quad y_{iD_{i,obs}1}, \ldots, y_{iD_{i,obs}m_{iD_{i,obs}}}, y_{iD_{i,obs}m_{iB_i}+1} < c, \\
& \ldots, y_{iD_{i,obs}n_{iD_{i,obs}}} < c; \\
& \quad y_{iD_{i,obs}+11} < c, \ldots, y_{iD_{i,obs}+1m_{iD_{i,obs}}} < c, \\
& \ldots, y_{iD_{i,obs}n_{iD_{i,obs}}} < c) \\
& \prod_{i=S_{obs}+1}^{S} L(y_{ijk} < c; j = 1, \ldots, D_i; k = 1, \ldots, n_{ij})
\end{aligned}$$

If Y_i denote the vector of observations of those from ith sire, Y_i and $Y_{i'}$ are independent for any $i \neq i'$. The log-likelihood of $Y = (Y_1, Y_2, \ldots, Y_s)$ becomes

$$\begin{aligned}
\log L(\mu, \sigma_0^2, \sigma_1^2, \sigma_2^2; Y) = & \sum_{i=1}^{S_{obs}} \log L(y_{i11}, \ldots, y_{i1m_{i1}}, y_{i1m_{i1}+1} < c, \ldots, y_{i1n_{i1}} < c; \\
& \qquad \ldots, \ldots \\
& \quad y_{iD_{i,obs}1}, \ldots, y_{iD_{i,obs}m_{iD_{i,obs}}}, y_{iD_{i,obs}m_{iB_i}+1} < c,
\end{aligned}$$

$$
\begin{aligned}
& \ldots, y_{iD_{i,obs}n_{iD_{i,obs}}} < c; \\
& \quad y_{iD_{i,obs}+11} < c, \ldots, y_{iD_{i,obs}+1m_{iD_{i,obs}}} < c, \\
& \ldots, y_{iD_{i,obs}n_{iD_{i,obs}}} < c) \\
+ & \sum_{i=S_{obs}+1}^{S} \log L(y_{ijk} < c; j = 1, \ldots, D_i; k = 1, \ldots, n_{ij})
\end{aligned}
$$

where c is a known constant.

In theory, the $\sigma_0^2, \sigma_1^2, \sigma_2^2$, and μ can be estimated by maximizing the above log-likelihood function. However, the typical component for sire i has the form

$$L(y_{i,obs.}, y_{i,mis.} < c) = \int_{-\infty}^{c} \cdots \int_{-\infty}^{c} f(y_{i,obs.}, y_{i,mis.}) dy_{i,mis.}$$

where $y_{i,obs.}$ is the vector of observed data for i while $y_{i,mis.}$ is the vector of missing data for i. Because the dimension of the integration is the length of $y_{i,mis.}$, the direct computation of $P(y_{i,obs.}, y_{i,mis.} < c)$ is practically impossible when there are more then a few missing data. The following proposition is necessary to transform $P(y_{i,obs.}, y_{i,mis.} < c)$ into a computable likelihood function.

Proposition 5 *Assume that Y_i is multivariate normal*

$$\begin{pmatrix} y_{i11} \\ y_{i12} \\ \vdots \\ y_{iD_in_{iD_i}} \end{pmatrix} \sim N(\mu, \Sigma_i),$$

where μ is a $\sum_{j=1}^{D_i} n_{ij} \times 1$ vector, and variance-covariance matrix Σ_i has form

$$\Sigma_i = \sigma_0^2 I + \sigma_1^2 J_i + \sigma_2^2 K_i.$$

Then the log-likelihood function can be written as

$$\log L = \sum_{i=1}^{S_{obs}} \log \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_0^2/2)$$

$$\{\prod_{j=1}^{D_{i,obs}}\int_{-\infty}^{\infty}\exp\{\frac{-1}{2}\sum_{k=1}^{m_{ij}}(\frac{y_{ijk}-\mu-\sigma_1 z_0-\sigma_2 z_j}{\sigma_0})^2\}$$

$$\Phi(w_j)^{n_i-m_i}\frac{1}{\sqrt{2\pi}}\exp(-z_j^2/2)(\sqrt{2\pi}\sigma_0)^{-m_i}dz_j$$

$$\prod_{j=D_{i,obs}+1}^{D_i}\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\exp(-z_j^2/2)\Phi(w_j)^{n_{ij}}dz_j\}dz_0$$

$$+\sum_{i=S_{obs}+1}^{S}\log\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\exp(-z_0^2/2)\prod_{j=1}^{D_i}\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\exp(-z_j^2/2)$$

$$\Phi(w_j)^{n_{ij}}dz_jdz_0. \tag{4.5}$$

where

$$w_j=\frac{c-\mu-\sigma_1 z_0-\sigma_2 z_j}{\sigma_0},\ \ hj=1,2,\ldots,D_i.$$

It is important to note that the likelihood now involves integration only over two dimensions rather than integration over the number of dimensions corresponding to the length of the missing data. Hence σ_0^2, σ_1^2, σ_2^2 and μ can be estimated by maximizing above function which is computable. For proof see Chapter 5.

4.4 Computation

Ordinarily, we must resort to an iterative numerical procedure to obtain a ML estimate of variance components. There are simple cases where the estimate can be found by analytical means (for example, balanced one-way random-effects model). The likelihood equations for full ML do not admit an explicit solution for all models (Hartley and Rao, 1967).

There are many iterative unmerical algorithms that can be regarded as candidates for computing ML estimates of variance components. Some were developed specifically for special cases, others are general procedures for the numerical solution of broad classes of constrained non-linear optimization problems. There is no single iterative numerical algorithm for ML estimation of variance components that will be best, or perhaps even satisfactory, for every application. An algorithm that requires relatively few computations to converge to a ML estimate in one setting may converge slowly or even fail to converge in another. In deciding which among available algorithms to try in a particular application, we must make some judgments about their computional requirements and their other properties as applied to a given setting (Harville, 1977).

It is noteworthy that the above likelihood function must be evaluated with two levels of integration, both of which have infinite range. This infinite range can cause numerical inaccuracies and a transformation is usually required. To obtain a better result, the following transformation has been applied to convert the infinite interval to the interval [-1, 1]:

$$\int_{-\infty}^{\infty} f(x)dx = \int_{0}^{1}[f(\frac{1-t}{t}) + f(\frac{-1+t}{t})]\frac{1}{t^2}dt$$

Gaussian quadrature method is used to evaluate the two levels of integration above.

To locate an ML estimate of variance components, we can use Newton-Raphson algorithm, steepest ascent algorithm, or Simplex method. Since the gradient of our likelihood function is too difficult to get (we confront with a situation that the derivative of the integration of a product of integration will be needed), I am forced to give up all the gradient procedures. The two methods which requires only funtion evaluations have been tried by using subroutines in NAG and IMSL, none of them works due to the nature of our likelihood function. The method I use here is global search of the parameter space. The maximum function value in a given parameter space is found by using a global search of the parameter space. It is not efficient in terms of the number of function evaluations that requires. However it can give the results with today's computer. A better maximization of likelihood function is needed.

For two-way nested model, the parameter space is a four dimensional space. The search strategy which I used in my program is

- I start at one-dimensional space. With three parameters (say, σ_1^2, σ_2^2 and μ) fixed and coarse grid, it is easy to find out the maximum of the function of σ_0^2;
- repeat above step for other three parameters. The four one-dimensional max imums will help us to localize a small four-dimensional space which usually include the four-dimensional maximum;
- with a fine grid and four-dimensional global search, the maximum function is usually located.

A typical problem as our motivating example needs roughly 65 function evaluations for localizing the final search space, and needs roughly 10000 function evaluations for finding the maximum.

A graphical data presentation programme for estimating σ_0^2, σ_1^2, σ_2^2 and μ was developed during this research on a personal computer systems taking advantage of 80-bit floating computation. With the aid of graphical data presentation, the user can visually determine the proper upper and lower bounds on the parameters and carry out an effective search.

By using the programme on an i486/50X PC, a modest problem like the example in Chapter 1 can be solved in few minutes.

For computing confidence intervals for variance components (say, for σ_0^2) by the likelihood ratio statistic, I will

- first use global search of the parameter space to find the estimates of varince components and μ as we developed before,
- replace the parameters in the likelihood function except σ_0^2 by the estimates,

- then calculate

$$L(\mu_{\hat{\sigma}_0^2}, \sigma_0^2, \hat{\sigma}^2_{1\sigma_0^2}, \hat{\sigma}^2_{2\sigma_0^2}; y) > L(\hat{\mu}, \hat{\sigma}_0^2, \hat{\sigma}_1^2, \hat{\sigma}_2^2; y) - \frac{\chi^2_{1,\alpha}}{2}$$

by the same global search routine (the search space is one dimension now).

4.5 examples

In this section, the preceding procedure is illustrated with five examples. The first example is the one introduced in chapter 1 as a motivating example. In chapter 1, we have seen the severe bias when we applied classical ANOVA and MLE methods to the largest 200 observations. Now we will use the same 200 data to estimate the variance components by our new procedure and we will see that the estimates are very close to true values. Based on simulated data. Example 2 gives the estimates by MLE and MIS for complete data sets. I want to use it to show the reliability of our programme. The box-plots for each component by three methods (ANOVA, MLE, and our method) are in the third example. As will be seen the MLE and ANOVA estimates have substantial bias for all four components while our estimates is approximately unbiased. In Example 4, we compute the confidence intervals. Example 5 shows the boxplot of the three methods with four different missing rates.

Example 1: The mating design had 7 sires with 3 dams nested within each sire, sixty offspring per female. 1260 offspring (sibs) are grown together in a common pool. At the end of the experiment all fish have been weighed, so their sizes (= growth rates because they are all the same age) are known. The largest 200 fish have been analyzed in the Gene Probe Lab so their parentage (sire and dam) is known. The parentage of the remaining 1060 fish is unknown. We have listed the estimates obtained by our method (MIS), compared with ANOVA and MLE in Table 4.1. As can be seen our method gives results very close to the true results.

Example 2: 3 simulated date sets with $\sigma_0^2 = \sigma_1^2 = \sigma_2^2 = 1.0$ are used (the number of sires, dams and offspring are randomly generated). The estimates obtained by MLE and MIS with no missing data are compared in Table 4.2.

We can see that MLE and MIS are very close when there is no missing data. Since the results of MLE with no missing data have been checked by SAS and BMDP, we can put trust in our programme. We also note that $L_{MLE_{max}} > L_{MIS_{max}}$ for the three simulated data sets. This indicates the error of numerical computation (evaluation the two levels of integration and optimization).

Example 3: 50 simulated data sets were generated with $\sigma_0^2 = 2.0$, $\sigma_1^2 = 1.0$ and $\sigma_2^2 = 0.5$, and $\mu = 0.0$ respectively. In each simulation, the number of sires, dams, and offspring were randomly generated and used to generate y_{ijk} for a complete set of data. After the complete set of data was generated, the largest 30% was used for variance component estimation by ANOVA, MLE, and our method. The box plot for each component by the three methods are shown in Figure 4.1. As can be seen the MLE and ANOVA estimates had substantial bias for all four components while our estimates are approximately unbiased.

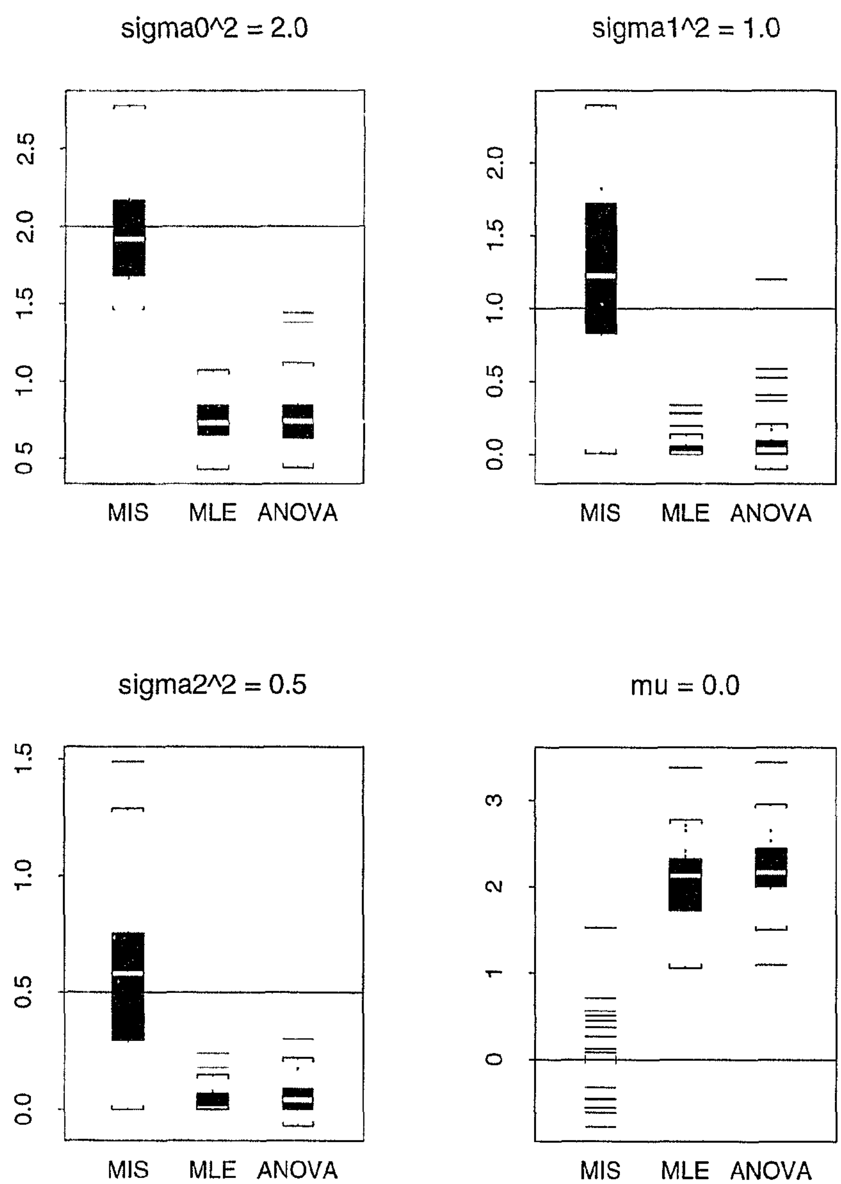

Fig 1.1. Box-plot for each component and μ by the three methods

It is not enough to report the point estimate. We have to know also what is the precision of the estimate. Generally, confidence interval formulates the precision of the estimate. The following example gives the confidence intervals for variance components with missing observations based on simulated data. The construction and computation of the confidence interval for variance components with missing observations are described in Chapter 5.

Example 4: We generated 25 groups of data using $\sigma_0^2 = 1.0$, $\sigma_1^2 = 1.0$, $\sigma_2^2 = 1.0$ and $\mu = 0.0$. In each simulation, the number of sires is randomly chosen between 5 and 10 (some of the long confidence intervals for σ_1^2 reflect the fact that the effective sample size for estimating σ_1^2 is small), the number of dams within ith sire is randomly chosen between 5 and 8 (overall sample size for dams is between 25 and 80), and the number of offspring is randomly chosen between 15 and 30 (overall sample size for offspring is between 375 and 2400). The largest 30% of the data were used for variance component estimation (both point and confidence interval) ($\alpha = 0.05$). The results are reported in Table 4.3. The coverage rates of our confidence intervals for σ_0^2, σ_1^2, σ_2^2, and μ are 100%, 96%, 96%, and 96% respectively.

Example 5: 12 data sets were generated with $\sigma_0^2 = 1.0$, $\sigma_1^2 = 1.0$, $\sigma_2^2 = 1.0$, and $\mu = 0.0$. In each simulation, 100 %, 70 %, 40 % and 10 % largest data were used for variance components estimation. Figure 4.2 shows the boxplots of $\hat{\mu}$ with different missing rates. We observe that bias of ANOVA and MLE increase as missing rate increase while our estimate is comparatively stable.

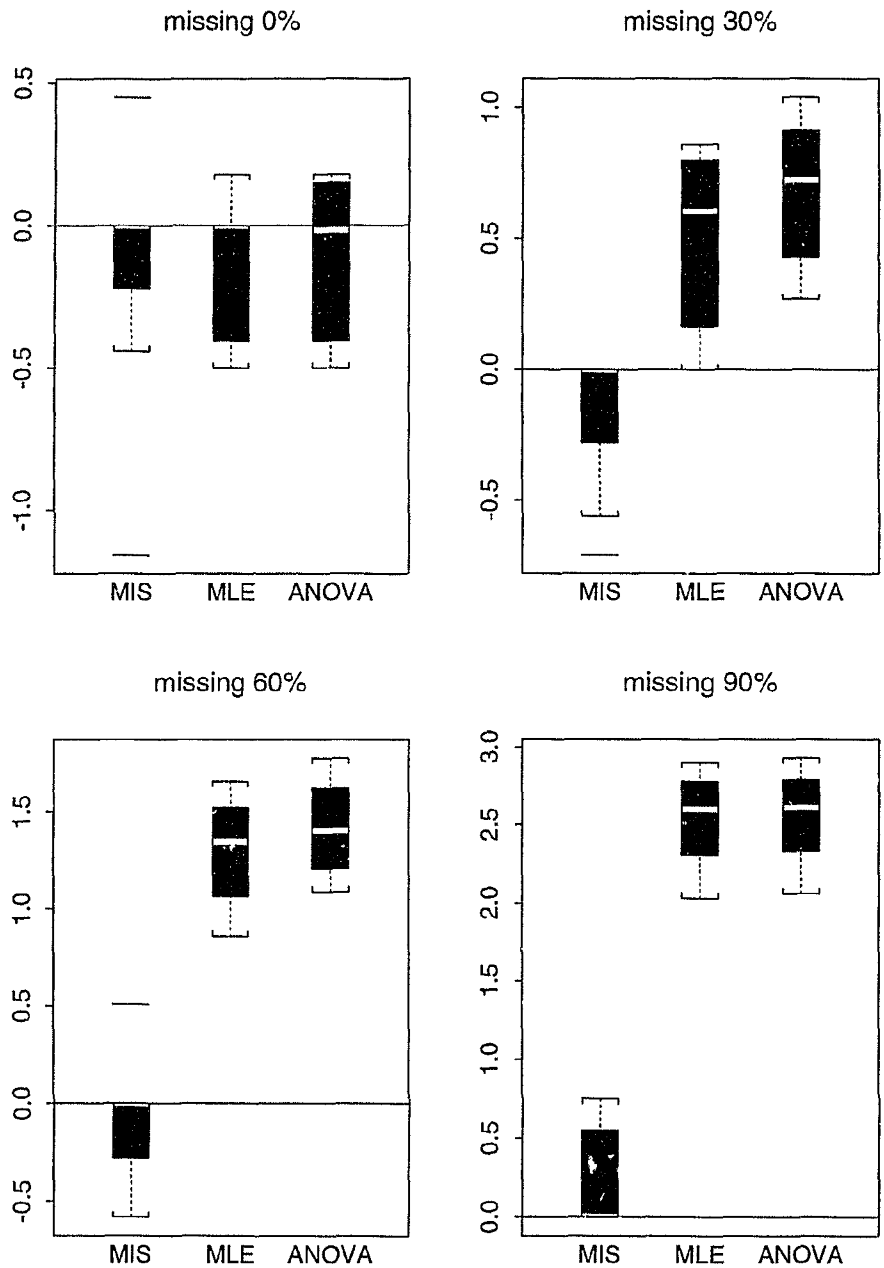

Fig 4.2 Box-plot with four different missing rates

Based on the simulation, we observed that

- When there is no missing observations, the results of three methods are quite similar.

- When missing rate increase from 30 % to 60 %, the bias of ANOVA and MLE increase. But our estimate does not seem to be affected.

Table 4.1: *Comparison of the three methods*

	σ_0^2	σ_1^2	σ_2^2	μ
True(1260)	100.75	29.05	18.70	28.17
ANOVA(200)	26.43	2.69	-1.019	47.10
MLE(200)	25.83	1.89	0.082	46.30
MIS(200)	95.00	26.60	18.05	28.00

Table 4.2: *Comparison of the three methods*

	MLE	MIS	MLE	MIS	MLE	MIS
σ_0^2	1.04	1.04	1.04	1.04	1.04	1.04
σ_1^2	1.98	2.06	2.07	2.50	0.92	0.67
σ_2^2	1.12	1.36	1.25	1.40	0.66	0.63
μ)	-0.40	-0.42	0.00	0.00	0.49	0.45
L_{max}	297.41	289.12	294.54	286.20	286.66	278.94

Table 4.3: *Point estimates and confidence intervals* $(\alpha = 0.05)$

	$\hat{\sigma}_0^2$	$\hat{\sigma}_1^2$	$\hat{\sigma}_2^2$	$\hat{\mu}$
1	1.04	0.68	1.10	-0.53
	(0.81, 1.34)	(0.02, 3.08)	(0.69, 3.71)	(1.16, 0.35)
2	1.27	5.34	0.59	0.28
	(0.98, 1.63)	(1.29, 10.15)	(0.31, 2.59)	(-2.48, 1.40)
3	0.875	1.16	1.372	-0.42
	(0.66, 1.15)	(0.32, 8.51)	(0.48, 3.61)	(-1.23, 0.68)
4	1.04	1.55	1.08	0.78
	(0.83, 1.37)	(0.29, 20.98)	(0.16, 3.39)	(-0.77, 2.31)
5	1.01	1.04	0.93	0.0
	(0.80, 1.32)	(0.0, 5.58)	(0.62, 3.93)	(-1.20, 0.63)
6	1.04	0.86	0.60	-0.87
	(0.79, 1.39)	(0.24, 5.83)	(0.53, 1.99)	(-1.60, 0.16)
7	1.08	1.97	2.67	1.0
	(0.88, 1.46)	(0.0, 15.38)	(0.85, 8.1)	(-0.27, 2.43)
8	1.03	1.50	0.44	-0.53
	(0.74, 1.30)	(0.56, 9.37)	(0.18, 1.76)	(-1.39, 0.80)
9	1.12	1.13	0.36	0.59
	(0.85, 1.51)	(0.37, 6.56)	(0.12, 1.38)	(-0.32, 1.59)
10	1.00	0.75	0.70	0.46
	(0.78, 1.30)	(0.25, 4.15)	(0.30, 2.01)	(-0.42, 1.21)
11	0.83	0.541	0.77	0.00
	(0.65, 1.13)	(0.07, 3.07)	(0.32, 1.96)	(-0.63, 0.68)
12	0.93	0.69	1.38	0.44
	(0.78, 1.29)	(0.19, 5.88)	(0.67, 4.78)	(-0.32, 1.44)
13	1.04	0.91	1.13	1.33
	(0.83, 1.36)	(0.26, 4.87)	(0.53, 2.03)	(0.60, 2.09)
14	0.88	0.91	3.35	-0.52
	(0.80, 1.24)	(0.46, 4.33)	(1.11, 7.57)	(-1.02, 0.68)
15	1.13	0.62	0.57	-0.36
	(0.79, 1.33)	(0.02, 4.21)	(0.27, 1.73)	(-1.04, 0.29)
16	0.87	0.88	1.37	0.49
	(0.67, 1.11)	(0.34, 3.69)	(0.81, 3.67)	(-0.23, 1.17)
17	0.92	1.61	2.82	1.11
	(0.70, 1.20)	(0.49, 9.19)	(0.78, 6.49)	(-0.63, 1.91)
18	0.99	2.15	0.96	-1.10
	(0.76, 1.28)	(0.57, 9.37)	(0.44, 3.31)	(-2.11, 0.23)
19	1.10	1.72	1.10	0.00
	(0.88, 1.44)	(0.40, 7.23)	(0.60, 4.08)	(-0.76, 1.26)

(continued)

	$\hat{\sigma}_0^2$	$\hat{\sigma}_1^2$	$\hat{\sigma}_2^2$	$\hat{\mu}$
20	0.98	2.24	0.98	0.38
	(0.79, 1.34)	(0.59, 14.19)	(0.32, 3.08)	(-0.81, 2.18)
21	1.15	2.05	0.85	-0.59
	(0.92, 1.51)	(0.39, 7.96)	(0.29, 2.24)	(-1.63, 0.49)
22	0.92	1.40	0.92	-1.01
	(0.78, 1.22)	(0.26, 4.59)	(0.51, 1.50)	(-1.46, 0.04)
23	1.03	2.71	0.36	0.42
	(0.80, 1.37)	(0.86, 18.20)	(0.26, 2.21)	(-1.47, 1.74)
24	0.98	0.77	0.78	0.56
	(0.80, 1.33)	(0.09, 4.72)	(0.29, 1.98)	(-0.43, 1.22)
25	1.00	0.66	0.85	0.00
	(0.77, 1.36)	(0.02, 4.31)	(0.57, 1.44)	(-0.19, 0.46)

Chapter 5

Properties of the Estimator

This chapter contains proofs for two propositions in Chapter 3 and 4 which are important in enabling us to transform the full likelihood functions into the computable functions. We also provide proofs for the existence of ML estimates of the parameters. Construction of confidence intervals is included in section 3.

5.1 Proofs of Proposition 2 and Proposition 5

We shall give the proof for Proposition 2 first. The proposition states that if

$$\begin{pmatrix} Y_{i1} \\ Y_{i2} \\ \vdots \\ Y_{in_i} \end{pmatrix} \sim N(\mu, \sigma_0^2 I_{n_i \times n_i} + \sigma_1^2 J_i),$$

then

$$L(y_{i1}, y_{i2}, \ldots, y_{im_i}, y_{im_i+1} < c, \ldots, y_{in_i} < c) \propto$$
$$\int_{-\infty}^{\infty} \sigma_0^{-m_i} \Phi(\frac{c - \mu - \sigma_1 z_0}{\sigma_0})^{n_i - m_i} \exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij} - \mu - \sigma_1 z_0)^2}{\sigma_0^2} - z_0^2]\} dz_0$$

where J_i is a $n_i \times n_i$ matrix consisting of $1's$ and L is defined as in Chapter 3.

PROOF. Let $Z_0, Z_{i1}, \ldots, Z_{in_i}$ denote independent normally distributed variables with zero means and unit variances, and define

$$Y_{ij} = \mu + \sigma_0 Z_{ij} + \sigma_1 Z_0.$$

Then the joint distribution of $Y_{i1}, Y_{i2}, \ldots, Y_{in_i}$ is a n_i-variate normal distribution with mean μ and variance-covariance matrix

$$Var \begin{pmatrix} Y_{i1} \\ Y_{i2} \\ \vdots \\ Y_{in_i} \end{pmatrix} = \sigma_0^2 I_{n_i \times n_i} + \sigma_1^2 J_i.$$

We define

$$L(\mu + \sigma_0 Z_{ij} + \sigma_1 Z_0 \mid Z_0 = z_0) =$$
$$\lim_{\Delta \to 0} P(\mu + \sigma_0 z_{ij} + \sigma_1 Z_0 \leq \mu + \sigma_0 Z_{ij} + \sigma_1 Z_0 \leq \mu + \sigma_0 (z_{ij} + \Delta) + \sigma_1 Z_0 \mid Z_0 = z_0).$$

Substituting Y_{ij} as $\mu + \sigma_0 Z_{ij} + \sigma_1 Z_0$, we obtain

$$\begin{aligned}
&L(Y_{i1} Y_{i2} \ldots, Y_{im_i}, Y_{im_i+1} < c, \ldots, Y_{in_i} < c) = \\
&\quad L(\mu + \sigma_0 Z_{i1} + \sigma_1 Z_0, \mu + \sigma_0 Z_{i2} + \sigma_1 Z_0, \ldots, \mu + \sigma_0 Z_{im_i} + \sigma_1 Z_0; \\
&\quad \mu + \sigma_0 Z_{im_i+1} + \sigma_1 Z_0 < c, \ldots, i\mu + \sigma_0 Z_{in_i} + \sigma_1 Z_0 < c) \\
&= \int_{-\infty}^{\infty} L(\mu + \sigma_0 Z_{i1} + \sigma_1 Z_0, \mu + \sigma_0 Z_{i2} + \sigma_1 Z_0, \ldots, \mu + \sigma_0 Z_{im_i} + \sigma_1 Z_0; \\
&\quad \mu + \sigma_0 Z_{im_i+1} + \sigma_1 Z_0 < c, \ldots, \mu + \sigma_0 Z_{in_i} + \sigma_1 Z_0 < c \mid Z_0 = z_0) f(z_0) dz_0 \\
&= \int_{-\infty}^{\infty} L(\mu + \sigma_0 Z_{i1} + \sigma_1 Z_0 \mid Z_0 = z_0) L(\mu + \sigma_0 Z_{i2} + \sigma_1 Z_0 \mid Z_0 = z_0) \\
&\quad \ldots L(\mu + \sigma_0 Z_{im_i} + \sigma_1 Z_0 \mid Z_0 = z_0) P(Z_{im_i+1} < \frac{c - \mu - \sigma_1 z_0}{\sigma_0} \mid Z_0 = z_0) \\
&\quad \ldots P(Z_{in_i} < \frac{c - \mu - \sigma_1 z_0}{\sigma_0} \mid Z_0 = z_0) f(z_0) dz_0 \\
&= \int_{-\infty}^{\infty} \prod_{j=1}^{m_i} f(Y_{ij} \mid Z_0 = z_0) \prod_{j=m_i+1}^{n_i} P(Z_{ij} < \frac{c - \mu - \sigma_1 z_0}{\sigma_0} \mid Z_0 = z_0) f(z_0) dz_0 \\
&= \int_{-\infty}^{\infty} \sigma_0^{-m_i} \Phi(\frac{c - \mu - \sigma_1 z_0}{\sigma_0})^{n_i - m_i} \exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i} (y_{ij} - \mu - \sigma_1 z_0)^2}{\sigma_0^2} + z_0^2]\} dz_0.
\end{aligned}$$

□

This proposition allows us to transform the full likelihood as a one-dimensional function which can be evaluated easily.

Let us consider the Proposition 5,

$$\textit{If} \qquad \begin{pmatrix} Y_{i11} \\ Y_{i12} \\ \vdots \\ Y_{iD_i n_{iD_i}} \end{pmatrix} \sim N(\mu, \Sigma_i),$$

then

$$P(y_{i,obs.}, y_{i,mis.} < c) = \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_0^2/2) h(z_0, m_i, n_i) dz_0$$

where

$$\begin{aligned} h(z_0, m_i, n_i) &= \prod_{j=1}^{D_i} \int_{-\infty}^{\infty} \exp\{\frac{-1}{2} \sum_{k=1}^{m_{ij}} (\frac{y_{ijk} - \mu - \sigma_1 z_0 - \sigma_2 z_j}{\sigma_0})^2\} \\ &\quad \Phi(\frac{c - \mu - \sigma_1 z_0 - \sigma_2 z_j}{\sigma_0})^{n_i - m_i} (\sqrt{2\pi}\sigma_0)^{-m_i} dz_j, \end{aligned}$$

$j = 1, 2, \ldots, D_i$,

$$\Sigma_i = \sigma_0^2 I + \sigma_1^2 J_i + \sigma_2^2 K_i,$$

J_i denotes a $\sum_{j=1}^{D_i} n_{ij} \times \sum_{j=1}^{D_i} n_{ij}$ matrix consisting of $1's$ and

$$K_i = \begin{pmatrix} 1_{i1}1'_{i1} & 0 & \ldots & 0 \\ 0 & 1_{i2}1'_{i2} & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & 1_{iD_i}1'_{iD_i} \end{pmatrix}.$$

where 1_{ij} is a vector of ones of length n_{ij}.

PROOF. Let $Z_0, Z_1, \ldots, Z_{D_i}, Z_{i11}, \ldots, Z_{iD_i n_{iD_i}}$ denote independent normally distributed variables with zero means and unit variances, and let

$$Y_{ijk} = \mu + \sigma_0 Z_{ijk} + \sigma_1 Z_0 + \sigma_2 Z_j.$$

Then the joint distribution of $Y_{i11}, Y_{i12}, \ldots, Y_{iD_in_{iD_i}}$ is a $\sum_{j=1}^{D_i} n_{ij}$ variate normal distribution with mean μ and the variance-covariance matrix

$$Var \begin{pmatrix} Y_{i11} \\ Y_{i12} \\ \vdots \\ Y_{iD_in_{iD_i}} \end{pmatrix} = \sigma_0^2 I_{n_i \times n_i} + \sigma_1^2 J_i \sigma_2^2 K_i.$$

Let $y_{i,obs.}$ represents the observed part of Y_i and $y_{i,mis.}$ denotes the missing values. As before we define

$$\begin{aligned} &L(\mu + \sigma_0 Z_{ijk} + \sigma_1 Z_0 + \sigma_2 Z_j \mid Z_0 = z_0, Z_j = z_j) = \\ &\quad \lim_{\Delta \to 0} P(\mu + \sigma_0 z_{ijk} + \sigma_1 Z_0 + \sigma_2 Z_j \leq \mu + \sigma_0 Z_{ijk} + \sigma_1 Z_0 + \sigma_2 Z_j \\ &\quad \leq \mu + \sigma_0 (z_{ijk} + \Delta) + \sigma_1 Z_0 + \sigma_2 Z_j \mid Z_0 = z_0, Z_j = z_j), \end{aligned}$$

and substitute Y_{ijk} as $\mu + \sigma_0 Z_{ijk} + \sigma_1 Z_0 + \sigma_2 Z_j$. The density of $(Y_{i,\text{obs.}}, Y_{i,\text{mis.}} < c)$ would be

$$\begin{aligned} &L(Y_{i,\text{obs.}} Y_{i,\text{mis.}} < c) = \\ &\quad L((\mu + \sigma_0 Z_{ijk} + \sigma_1 Z_0 + \sigma_2 Z_j)_{i,\text{obs.}}, (\mu + \sigma_0 Z_{ijk} + \sigma_1 Z_0 + \sigma_2 Z_j)_{i,\text{mis.}} < c) \\ &= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} L((\mu + \sigma_0 Z_{ijk} + \sigma_1 Z_0 + \sigma_2 Z_j)_{i,\text{obs.}}, \\ &\quad (\mu + \sigma_0 Z_{ijk} + \sigma_1 Z_0 + \sigma_2 Z_j)_{i,\text{mis.}} < c \mid Z_0, \ldots, Z_{D_i}) f(Z_0, \ldots, Z_{D_i}) dz_0, \ldots, dz_{D_i} \\ &= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \prod_{j=m_i+1}^{n_{iD_i}} P(Z_{ijk} < \frac{c - \mu - \sigma_1 z_0 - \sigma_2 z_j}{\sigma_0} \mid Z_0, \ldots, Z_{D_i}) \\ &\quad f(Z_0, \ldots, Z_{D_i}) \prod_{j=1}^{m_i} f(Y_{ijk} \mid Z_0, \ldots, Z_{D_i}) dz_0, \ldots, dz_{D_i} \\ &\propto \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i} (y_{ijk} - \mu - \sigma_1 z_0 - \sigma_2 z_j)^2}{\sigma_0^2} + z_0^2 + z_j^2]\} \\ &\quad \sigma_0^{-m_i} \Phi(w_j)^{n_i - m_i} dz_0, \ldots, dz_{D_i} \\ &= \int_{-\infty}^{\infty} \exp(-z_0^2/2) \prod_{j=1}^{D_i} \int_{-\infty}^{\infty} \exp(-z_j^2/2) \exp\{\frac{-1}{2} \sum_{k=1}^{m_{ij}} (\frac{y_{ijk} - \mu - \sigma_1 z_0 - \sigma_2 z_j}{\sigma_0})^2\} \\ &\quad \Phi(w_j)^{n_i - m_i} (\sqrt{2\pi}\sigma_0)^{-m_i} dz_j dz_0 \end{aligned}$$

□

This proposition enables us to have a likelihood which involves integration only over two dimension rather than integration over the number of dimensions corresponding to the length of the missing data. This is the key step of our method.

5.2 The Existence of ML estimate

ML estimates may not always exist (Rao and Kieffe, 1988). However, we have proved

Theorem 1 *For model (4.5), there is a ML estimate.*

PROOF. Let $\theta = (\sigma_0, \sigma_1, \sigma_2, \mu) \in [0, \infty)^3 \times (-\infty, \infty)$ and

$$\begin{aligned}
f_\theta &= L(\mu, V; Y) \\
&= \prod_{i=1}^{S_{obs}} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_0^2/2) \\
&\quad \{ \prod_{j=1}^{D_{i,obs}} \int_{-\infty}^{\infty} \exp\{\frac{-1}{2} \sum_{k=1}^{m_{ij}} (\frac{y_{ijk} - \mu - \sigma_1 z_0 - \sigma_2 z_j}{\sigma_0})^2\} \\
&\quad \Phi(w_j)^{n_i - m_i} \frac{1}{\sqrt{2\pi}} \exp(-z_j^2/2)(\sqrt{2\pi}\sigma_0)^{-m_i} dz_j \\
&\quad \prod_{j=D_{i,obs}+1}^{D_i} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_j^2/2)\Phi(w_j)^{n_{ij}} dz_j\} dz_0 \\
&\quad \prod_{i=S_{obs}+1}^{S} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_0^2/2) \prod_{j=1}^{D_i} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_j^2/2)\Phi(w_j)^{n_{ij}} dz_j dz_0.
\end{aligned}$$

where $\Phi(w) = \int_{-\infty}^{w} \phi(x)dx$ and $w_j = \frac{c - \mu - \sigma_1 z_0 - \sigma_2 z_j}{\sigma_0}$.

We are going to show that if $\hat{\theta}$ is the MLE of f_θ, then $\exists N < \infty$ and $\hat{\theta} \in [0, N]^3 \times [-N, N]$.

$f_\theta \geq 0$ implies that $\exists \delta > 0$ and $\exists \theta' \in [0, \infty)^3 \times (-\infty, \infty)$ s.t. $f_{\theta'} \geq \delta$.

Since $\Phi(w)$ is bounded, as θ goes to infinity $\exp\{\frac{-1}{2}\sum_{k=1}^{m_{ij}}(\frac{y_{ijk} - \mu - \sigma_1 z_0 - \sigma_2 z_j}{\sigma_0})^2\}$ is bounded and $\frac{1}{\sqrt{2\pi}}\exp(-z_j^2/2)(\sqrt{2\pi}\sigma_0)^{-m_i}$ goes to zero. We can obtain that $\lim_{\theta \to \infty} f_\theta$

0. Taking any $\epsilon = \delta$, we should have

$$\exists N, \forall \theta \in ([0, N]^3 \times [-N, N])^c, f_\theta \le \epsilon = \delta.$$

It is easy to see that f_θ is continuous on $[0, N]^3 \times [-N, N]$. Therefore we know that f_θ takes on its maximum value at least once in $[0, N]^3 \times [-N, N]$ by the result in any calculus text book(Swokowski, 1979).

The uniqueness of ML estimates of the parameters has not well proved yet, but we have showed two properties which related to the convexity of the likelihood f_θ. I hope these will help us to establish uniqueness in further research.

- f_θ is closed: Suppose

  $$f_{\theta_n} \longrightarrow f_0, \forall \{\theta_n\} \subset [0, N]^3 \times [-N, N]$$

  We can show that we have $f_0 \in \Gamma$ as following.

  Since $[0, N]^3 \times [-N, N]$ is closed, we always can find a subsequence $\{\theta_{n_k}\}$ s.t.

  $$\theta_{n_k} \longrightarrow \theta_0 \in [0, N]^3 \times [-N, N]$$

  and moreover

  $$\begin{aligned} & \lim_{i \to \infty} f_{\theta_{n_i}} = \\ &= \lim_{i \to \infty} \{ \prod_{t=1}^{S_{obs}} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_0^2/2) \\ & \quad h(\theta_{n_i}; z_0, m_i, n_i) dz_0 \\ &= \prod_{t=1}^{S_{obs}} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_0^2/2) \\ & \quad h(\lim_{i \to \infty} \theta_{n_i}; z_0, m_i, n_i) dz_0 \\ &= f_{\theta_0} = f_0 \end{aligned}$$

  where

  $$h(\theta_{n_i}; z_0, m_i, n_i) = \{ \prod_{j=1}^{D_{i,obs}} \int_{-\infty}^{\infty} \exp\{ \frac{-1}{2} \sum_{k=1}^{m_{ij}} (\frac{y_{ijk} - \mu - \sigma_1 z_0 - \sigma_2 z_j}{\sigma_0})^2 \}$$

$$\Phi(w_j)^{n_i - m_i} \frac{1}{\sqrt{2\pi}} \exp(-z_j^2/2)(\sqrt{2\pi}\sigma_0)^{-m_i} dz_j$$

$$\prod_{j=D_{i,obs}+1}^{D_i} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_j^2/2)\Phi(w_j)^{n_{ij}} dz_j\} dz_0$$

$$\prod_{i=S_{obs}+1}^{S} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_0^2/2)$$

$$\prod_{j=1}^{D_i} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_j^2/2)\Phi(w_j)^{n_{ij}} dz_j.$$

- f_θ is bounded: Notice that

$$\begin{aligned}
&\Phi(w_j)^{n_i - m_i}(\sqrt{2\pi}\sigma_0)^{-m_i} \exp\{\frac{-1}{2}\sum_{k=1}^{m_{ij}}(\frac{y_{ijk} - \mu - \sigma_1 z_0 - \sigma_2 z_j}{\sigma_0})^2\} \\
\leq\ & (\sqrt{2\pi}\sigma_0)^{-m_i} \exp\{\frac{-1}{2}\sum_{k=1}^{m_{ij}}(\frac{y_{ijk} - \mu - \sigma_1 z_0 - \sigma_2 z_j}{\sigma_0})^2\}.
\end{aligned}$$

Hence

$$\begin{aligned}
&\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_0^2/2) \\
&\quad \{\prod_{j=1}^{D_{i,obs}} \int_{-\infty}^{\infty} \exp\{\frac{-1}{2}\sum_{k=1}^{m_{ij}}(\frac{y_{ijk} - \mu - \sigma_1 z_0 - \sigma_2 z_j}{\sigma_0})^2\} \\
&\quad \Phi(w_j)^{n_i - m_i} \frac{1}{\sqrt{2\pi}} \exp(-z_j^2/2)(\sqrt{2\pi}\sigma_0)^{-m_i} dz_j \\
&\quad \prod_{j=D_{i,obs}+1}^{D_i} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_j^2/2)\Phi(w_j)^{n_{ij}} dz_j\} dz_0 \\
\leq\ & \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_0^2/2) \prod_{j=1}^{D_i} \int_{-\infty}^{\infty} (\sqrt{2\pi}\sigma_0)^{-m_i} \\
&\quad \exp\{\frac{-1}{2}\sum_{k=1}^{m_{ij}}(\frac{y_{ijk} - \mu - \sigma_1 z_0 - \sigma_2 z_j}{\sigma_0})^2\} dz_j dz_0.
\end{aligned}$$

Also note that the function at right hand side is the likelihood with complete data. It then follows that

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp(-z_0^2/2) \prod_{j=1}^{D_i} \int_{-\infty}^{\infty} (\sqrt{2\pi}\sigma_0)^{-m_i}$$

$$
\begin{aligned}
&\exp\{\frac{-1}{2}\sum_{k=1}^{m_{ij}}(\frac{y_{ijk}-\mu-\sigma_1 z_0-\sigma_2 z_j}{\sigma_0})^2\}dz_j dz_0 \\
= \;& (2\pi)^{\frac{-\sum_j m_{ij}}{2}}|\Sigma_i|^{-1/2}\exp\{\frac{-1}{2}(Y_i-\mu_i)'\Sigma_i^{-1}(Y_i-\mu_i)\}
\end{aligned}
$$

where $\Sigma_i = \sigma_0^2 I_{n_i\times n_i} + \sigma_1^2 J_i + \sigma_2^2 K_i$. By Rao and Kleffe (1988),

$$
(2\pi)^{\frac{-\sum_j m_{ij}}{2}}|\Sigma_i|^{-1/2}\exp\{\frac{-1}{2}(Y_i-\mu_i)'\Sigma_i^{-1}(Y_i-\mu_i)\}.
$$

is bounded. Thus

$$
\begin{aligned}
&\sum_{i=1}^{S_{obs}}\log\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\exp(-z_0^2/2) \\
&\{\prod_{j=1}^{D_{i,obs}}\int_{-\infty}^{\infty}\exp\{\frac{-1}{2}\sum_{k=1}^{m_{ij}}(\frac{y_{ijk}-\mu-\sigma_1 z_0-\sigma_2 z_j}{\sigma_0})^2\} \\
&\Phi(w_j)^{n_i-m_i}\frac{1}{\sqrt{2\pi}}\exp(-z_j^2/2)(\sqrt{2\pi}\sigma_0)^{-m_i}dz_j \\
&\prod_{j=D_{i,obs}+1}^{D_i}\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\exp(-z_j^2/2)\Phi(w_j)^{n_{ij}}dz_j\}dz_0 \\
&+\sum_{i=S_{obs}+1}^{S}\log\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\exp(-z_0^2/2)\prod_{j=1}^{D_i}\int_{-\infty}^{\infty}\frac{1}{\sqrt{2\pi}}\exp(-z_j^2/2) \\
&\Phi(w)^{n_{ij}}dz_j dz_0
\end{aligned}
$$

is bounded as well.

By an argument similar to that of Theorem 1, we have a simpler case

Theorem 2 *For model (3.3), there is a ML estimate.*

5.3 Confidence Intervals

Calculating point estimates for variance components with missing data is usually not enough. We must give the user an idea of the precision or possible error of the estimates. In many estimation situations, it is of substantial interest to compute confidence intervals for parameters.

Approximate confidence intervals for variance components can be obtained from the likelihood-ratio statistic. Let us assume that $(Y_1, \ldots, Y_S)$ is a two-way nested sample where Y_i denote the vector of observations of those from ith sire. Y_i is a $\sum_{j=1}^{D_i} n_{ij}$ dimensional normal distribution with mean μ_i and nonsingular covariance matrix

$$\sigma_0^2 I_i + \sigma_1^2 J_i + \sigma_2^2 K_i.$$

Then the likelihood is given by expression (4.5) of Chapter 4. If we denote (4.5) as $l(\theta; Y)$ where

$$\theta = (\mu, \sigma_0^2, \sigma_1^2, \sigma_2^2),$$

the likelihood-ratio statistic will be

$$\lambda = \frac{l(\theta; Y)_{max_{\theta \in \omega}}}{l(\theta; Y)_{max_{\theta \in \Omega}}}$$

where Ω is the parameter space and ω is the subspace corresponding to the null hypothesis. The large-sample distribution theory of likelihood statistics implies that $-2ln\lambda$ has approximate distribution $\chi^2_{m(m+1)/2}$ where m is the number of parameters tested (McCullagh and Nelder, 1989).

Let $l(\theta; Y)_{max_{\theta \in \Omega}}$ be $l(\hat{\mu}, \hat{\sigma_0^2}, \hat{\sigma_1^2}, \hat{\sigma_2^2}; Y)$. For fixed σ_0^2,

$$l(\mu, \sigma_0^2, \sigma_1^2, \sigma_2^2; y)$$

is maximized at $(\hat{\mu}_{\sigma_0^2}, \sigma_0^2, \hat{\sigma}^2_{1\sigma_0^2}, \hat{\sigma}^2_{2\sigma_0^2})$. Thus

$$2l(\hat{\mu}, \hat{\sigma_0^2}, \hat{\sigma_1^2}, \hat{\sigma_2^2}; y) - 2l(\hat{\mu}_{\sigma_0^2}, \sigma_0^2, \hat{\sigma}^2_{1\sigma_0^2}, \hat{\sigma}^2_{2\sigma_0^2}; y) \sim (\chi_{1,\alpha})^2 + o(n^{-1}).$$

These approximations are often quite accurate for small values of S even when Normal approximations for parameter estimates are unsatisfactory (McCullagh and Nelder, 1989).

The set of all σ_0^2-values satisfying

$$2l(\hat{\mu}, \hat{\sigma_0^2}, \hat{\sigma_1^2}, \hat{\sigma_2^2}; y) - 2l(\hat{\mu}_{\sigma_0^2}, \sigma_0^2, \hat{\sigma^2_1}_{\sigma_0^2}, \hat{\sigma^2_2}_{\sigma_0^2}; y) \leq \chi^2_{1,\alpha}$$

is an approximate $100(1-\alpha)\%$ confidence set for σ_0^2.

Similarly we could get approximate $100(1-\alpha)\%$ confidence sets for μ, σ_1^2 and σ_2^2.

Chapter 6

Robust Procedures

This chapter provides robust procedures for variance components estimation with missing data from the one-way model.

6.1 The Need for Robust Statistics

As we mentioned in Chapter 1, statistical inferences are based only in part upon the observations. An equally important base is formed by prior assumptions about the underlying situation. There are explicit or implicit assumptions about randomness and independence, about distributional models, perhaps prior distributions for some unknown parameters, and so on. These assumptions are not supposed to be exactly true, but we would like to ensure that a minor deviation from the assumptions causes only a small change in the final conclusions. Robustness means insensitivity to small deviations from the assumptions.

In the one-way model, we have

$$y_{ij} = \mu + \alpha_i + \epsilon_{ij}, \begin{cases} i = 1, 2, \ldots, A \\ j = 1, 2, \ldots, n_i \end{cases}$$

where α_i and ϵ_{ij} are assumed to be independent $N(0, \sigma_1^2)$ and $N(0, \sigma_0^2)$. Both the random effects α_i and the random errors ϵ_{ij} can be contaminated. In principle, robust procedures should be able to withstand contamination in both components (Rocke 1983, Fellner 1986). Other departures from the underlying model, such as non-additivity or lack of independence of errors and/or random effects, will not be considered here.

The following simulated examples illustrate the need for robustness for our estimates.

Example 6.1: Assume that data is collected from a one-way model with 5 classes and 8 observation each class. In the simulation, we set mean $\mu = 0$, $\alpha_i \sim N(0, 1)$, and $\epsilon_{ij} \sim 0.95N(0, 1) + 0.05N(0, 50)$, respectively. α_i and ϵ_{ij} are independent. The estimates by ANOVA, MLE and MLE_{mis} with complete data are summarized in table 6.1.

	σ_0^2	σ_1^2	μ
ANOVA	4.7694	2.5854	-0.0388
MLE	4.7973	1.9174	-0.0348
MLE_{mis}	4.7173	2.3957	-0.0206

Table 6.1: $\sigma_0^2 = 1.0$, $\sigma_1^2 = 1.0$ *and* $\mu = 0.0$ *with complete data*

For complete data (40 observations), it can be seen that ANOVA, MLE and our estimate MLE_{mis} could give very poor estimates when ϵ_{ij} has a contamination distribution.

Example 6.2. Taking the same data set in Example 6.1 and only using data which are great than 0.0 (the largest 22 observations have been used), we list the results by three methods in the following table 6.2.

The numerical results in table 6.2 shows that with missing data the three methods all could give very poor estimates when the ϵ_{ij} are contaminated.

Table 6.2: $\sigma_0^2 = 1.0$, $\sigma_1^2 = 1.0$ *and* $\mu = 0.0$ *with data which are greater than 0.0*

	σ_0^2	σ_1^2	μ
ANOVA	3.2895	0.1251	1.3242
MLE	3.0862	0.1702	1.2178
MLE_{mis}	3.0441	0.1506	0.0100

6.2 The Influence Function

The influence function gives us a precise idea of how the estimator responds to a small amount of contamination at any point. Those estimators which are very sensitive to the form of F will be most influenced by small amounts of contamination.

Formally, consider observations $x_1, \ldots, x_n$ from an underlying density $f_\theta(x)$ where $\theta = (\theta_1, \ldots, \theta_p)$. Note that the $x_i's$ may be univariate or multivariate. Any estimate T_n is defined by a minimum problem of the form

$$\rho(x_i; T_n) = min!$$

or by an implicit equation

$$\sum_i \psi_k(x_i; T_n) = 0 \quad k = 1, \ldots, p$$

where ρ is an arbitrary function and $\psi(x;\theta) = (\partial/\partial\theta)\rho(x;\theta)$, is called an M-estimator (Field, 1982).

In the setting of interest for our 1-way problem, we consider observations $Y_1, \ldots, Y_A$ from a multivariate normal distribution $f_\theta(Y_i)$ where $\theta = (\mu, \sigma_0^2, \sigma_1^2)$ and $i = 1, \ldots, A$. As we have developed in Chapter 3, the MLE_{mis} estimates $T_n = (\hat{\sigma_0^2}, \hat{\sigma_1^2}, \hat{\mu})$ are the solutions of

$$\begin{aligned}\log L(\theta;Y) &= \sum_{i=1}^{A}\log L(\theta;Y_i)\\ &= \sum_{i=1}^{A_{obs}}\log\int_{-\infty}^{\infty}\sigma_0^{-m_i}\Phi(w)^{n_i-m_i}\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 y_0)^2}{\sigma_0^2}+z_0^2]\}dz_0\\ &+ \sum_{i=A_{obs}+1}^{A}\log\int_{-\infty}^{\infty}\Phi(w)^{n_i}\exp(-\frac{z_0^2}{2})dz_0 = min! \end{aligned} \tag{6.1}$$

where ρ is $\log L(\theta;Y_i)$, $\psi_k(\theta;Y_i) = (\partial/\partial\theta_k)\log L(\theta;Y_i)$, $k = 1,2,3$, $w = \frac{c-\mu-\sigma_1 z_0}{\sigma_0}$. Therefore the variance components estimate with missing data by our method T_n is an M-estimator.

Since the influence function of an M-estimator is proportional to ψ (Staudte and Sheather, 1990), it is easier to study the influence function of an M-estimator through its score function ψ. The relationship

$$IF(x;F,T) = \frac{\psi(x;T(F))}{-\int(\partial/\partial\theta)\psi(x;T(F))F(dx)}$$

shows that an M-estimator can in principle be defined by choice of ψ function to have desirable properties of efficiency and robustness. Robustness would be achieved by choosing ψ that is smooth and bounded to reduce the influence of extrem observations.

To examine the influence function of our MLE_{mis} estimator, we derive ψ_k from the log-likelihood function of the 1-way model with missing data. Each term in the function (3.3) of Chapter 3 is of one of two forms. Either

$$\log\int_{-\infty}^{\infty}\sigma_0^{-m_i}\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2}+z_0^2]\}dz_0 \tag{6.2}$$

or

$$\log\int_{-\infty}^{\infty}\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i}\exp(-z_0^2/2)dz_0 \tag{6.3}$$

We denote term (6.2) as L_{fi} and term (6.3) as $L_{\Phi i}$. Note that $L_{\Phi i}$ is independent of y_{ij} and also is bounded by $\log\sqrt{2\pi}$. When we look at the influence function as $y_{ij}\to\infty$, we only need to consider L_{fi} for the $\lim_{y_{ij}\to\infty}\psi_k(Y_i;\theta)$.

We begin with the derivative

$$
\begin{aligned}
\psi_{\sigma_0^2} = \frac{\partial L_{fi}(\theta; Y_i)}{\partial \sigma_0^2} = & \\
& \{\int_{-\infty}^{\infty} (d\sigma_0^{-m_i}/d\sigma_0^2)\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\sum_{j=1}^{m_i}(\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0})^2 + z_0^2]\}dz_0\} \\
/ & \{\int_{-\infty}^{\infty} \sigma_0^{-m_i}\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\sum_{j=1}^{m_i}(\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0})^2 + z_0^2]\}dz_0\} \\
+ & \{\int_{-\infty}^{\infty} \sigma_0^{-m_i}(d\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}/d\sigma_0^2) \\
\times & \exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2} + z_0^2]\}dz_0\} \\
/ & \{\int_{-\infty}^{\infty} \sigma_0^{-m_i}\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2} + z_0^2]\}dz_0\} \\
+ & \{\int_{-\infty}^{\infty} \sigma_0^{-m_i}\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i} \\
\times & (d\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2} + z_0^2]\}/d\sigma_0^2)dz_0\} \\
/ & \{\int_{-\infty}^{\infty} \sigma_0^{-m_i}\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2} + z_0^2]\}dz_0\} \\
= & g_1 + g_2 + g_3
\end{aligned}
$$

where

$$g_1 = \frac{-m_i}{2\sigma_0^2}L_{fi}/L_{fi} = \frac{-m_i}{2\sigma_0^2},$$

$$
\begin{aligned}
g_2 = & \\
& \{\int_{-\infty}^{\infty}(n_i - m_i)\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i-1}\phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0}) \\
& (-\frac{c-\mu-\sigma_1 z_0}{2\sigma_0^3})\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2} + z_0^2]\}dz_0\} \\
/ & \{\int_{-\infty}^{\infty}\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2} + z_0^2]\}dz_0\},
\end{aligned}
$$

and

$$g_3 =$$
$$\{\int_{-\infty}^{\infty} \Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i} \exp\{-\frac{1}{2}[\sum_{j=1}^{m_i}(\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0})^2 + z_0^2]\}$$
$$(\frac{1}{\sigma_0}\sum_{j=1}^{m_i}(\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0^2})^2 dz_0\}$$
$$/ \quad \{\int_{-\infty}^{\infty} \Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i} \exp\{-\frac{1}{2}[\sum_{j=1}^{m_i}(\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0^2})^2 + z_0^2]\}dz_0\}.$$

We deal with each term separately. It is obvious that $g_1 = \frac{-m_i}{2\sigma_0^2}$ is bounded. Let $g(\theta, Y_i, z_0) = \frac{1}{2}[\sum_{j=1}^{m_i}(\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0^2})^2 + z_0^2]$ and $\frac{c-\mu-\sigma_1 z_0}{\sigma_0} = t$. Then

$$g_2 = \{(n_i - m_i)(\frac{\sigma_0}{\sigma_1})\{\int_{-\infty}^{\infty} \Phi(t)^{n_i-m_i-1}\phi(t)$$
$$\times(\frac{t}{2\sigma_0^2})\exp(-g(\theta, Y_i, t))dt\}$$
$$/ \quad \{\int_{-\infty}^{\infty} \Phi(t)^{n_i-m_i}\exp(-g(\theta, Y_i, t)dt\}$$

Using $\Phi(t) \leq 1$ and $\exp(-g(\theta, Y_i, t)) \leq 1$, we obtain

$$0 \leq g_2 \leq \frac{\frac{(n_i-m_i)}{2\sigma_1\sigma_0}\int t\phi(t)dt}{\int \Phi(t)^{n_i-m_i}\exp(-g(\theta, Y_i, t))dt}.$$

The RHS is bounded as $y_{ij} \to \infty$, so that g_2 is bounded.

We now consider g_3. Note that

$$\begin{aligned} \frac{1}{\sigma_0}\sum_{j=1}^{m_i}(\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0^2})^2 &= \frac{1}{\sigma_0}\sum_{j=1}^{m_i}(\frac{y_{ij}-c}{\sigma_0^2}+t)^2 \\ &= \frac{1}{\sigma_0}\sum_{j=1}^{m_i}((\frac{y_{ij}-c}{\sigma_0^2})^2 + t^2 + 2t(\frac{y_{ij}-c}{\sigma_0^2})). \end{aligned}$$

Now the first term of g_3 becomes

$$\frac{\frac{1}{\sigma_0}\sum_{j=1}^{m_1}(\frac{y_{ij}-c}{\sigma_0^2})^2\int_{-\infty}^{\infty}\Phi(t)\exp(-g(\theta, Y_i, t))dt}{\int_{-\infty}^{\infty}\Phi(t)\exp(-g(\theta, Y_i, t))dt}.$$

Further reduction of the first term of g_3 gives

$$\frac{1}{\sigma_0}\sum_{j=1}^{m_1}(\frac{y_{ij}-c}{\sigma_0^2})^2.$$

As $y_{ij}\to\infty$, we obtain

$$\lim_{y_{ij}\to\infty}\frac{1}{\sigma_0}\sum_{j=1}^{m_1}(\frac{y_{ij}-c}{\sigma_0^2})^2=\infty.$$

We can show (as we did for g_2) that

$$\frac{m_i\int_{-\infty}^{\infty}t^2\Phi(t)\exp(-g(\theta,Y_i,t))dt}{\sigma_0\int_{-\infty}^{\infty}\Phi(t)\exp(-g(\theta,Y_i,t))dt}$$

and

$$\frac{2\sum_{j=1}^{m_i}(\frac{y_{ij}-c}{\sigma_0^2})\int_{-\infty}^{\infty}t\Phi(t)\exp(-g(\theta,Y_i,t))dt}{\int_{-\infty}^{\infty}\Phi(t)\exp(-g(\theta,Y_i,t))dt}$$

are bounded when y_{ij} goes to infinity.

It follows that $\lim_{y_{ij}\to\infty}g_3=\infty$, so that the influence function for $\hat{\sigma}_0^2$ is not bounded.

Next consider $\lim_{y_{ij}\to\infty}\psi_{\sigma_1^2}$.

$$\begin{aligned}
\frac{\partial\log f(\theta;Y_i)}{\partial\sigma_1^2} &= \\
&\{\int_{-\infty}^{\infty}(d\Phi(\frac{c-\mu-\sigma_1z_0}{\sigma_0})^{n_i-m_i}/d\sigma_1^2)\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1z_0)^2}{\sigma_0^2}+z_0^2]\}dz_0\} \\
/ &\quad \{\int_{-\infty}^{\infty}\Phi(\frac{c-\mu-\sigma_1z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1z_0)^2}{\sigma_0^2}+z_0^2]\}dz_0\} \\
+ &\quad \{\int_{-\infty}^{\infty}\Phi(\frac{c-\mu-\sigma_1z_0}{\sigma_0})^{n_i-m_i}(d\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1z_0)^2}{\sigma_0^2}+z_0^2]\}/d\sigma_1^2)dz_0\} \\
/ &\quad \{\int_{-\infty}^{\infty}\Phi(\frac{c-\mu-\sigma_1z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1z_0)^2}{\sigma_0^2}+z_0^2]\}dz_0\} \\
= &\quad v_1+v_2
\end{aligned}$$

where

$$
\begin{aligned}
v_1 = & \\
& \{\int_{-\infty}^{\infty} ((n_i - m_i)\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i-1}\phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})(-\frac{z_0}{\sigma_0}) \\
& \exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2}+z_0^2]\}dz_0\} \\
/ & \quad \{\int_{-\infty}^{\infty} \Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2}+z_0^2]\}dz_0\},
\end{aligned}
$$

and

$$
\begin{aligned}
v_2 = & \\
& \{\int_{-\infty}^{\infty} \Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\sum_{j=1}^{m_i}\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0^2})^2+z_0^2]\} \\
& \frac{z_0}{\sigma_0}(\sum_{j=1}^{m_i}(\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0}))dz_0\} \\
/ & \quad \{\int_{-\infty}^{\infty} \Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\sum_{j=1}^{m_i}(\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0^2})^2+z_0^2]\}dz_0\}.
\end{aligned}
$$

As before, we can show that v_1 is bounded and v_2 goes to infinity as y_{ij} increases. Hence our $\hat{\sigma}_1^2$ does not have a bounded influence function.

For $\hat{\mu}$, we find

$$
\begin{aligned}
\frac{\partial \log f(\theta; Y_i)}{\partial \mu} = & \\
& \{\int_{-\infty}^{\infty} (d\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}/d\mu)\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2}+z_0^2]\}dz_0\} \\
/ & \quad \{\int_{-\infty}^{\infty} \Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2}+z_0^2]\}dz_0\} \\
+ & \quad \{\int_{-\infty}^{\infty} \Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}(d\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2}+z_0^2]\}/d\mu)dz_0\} \\
/ & \quad \{\int_{-\infty}^{\infty} \Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2}+z_0^2]\}dz_0\} \\
= & \quad w_1 + w_2
\end{aligned}
$$

where

$$w_1 =$$

$$\{\int_{-\infty}^{\infty}((n_i - m_i)\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i-1}\phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})(\frac{1}{\sigma_0})$$

$$\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2} + z_0^2]\}dz_0\}$$

$$/ \quad \{\int_{-\infty}^{\infty}\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\frac{\sum_{j=1}^{m_i}(y_{ij}-\mu-\sigma_1 z_0)^2}{\sigma_0^2} + z_0^2]\}dz_0\},$$

and

$$w_2 =$$

$$\{\int_{-\infty}^{\infty}\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp(-\frac{1}{2}[\sum_{j=1}^{m_i}\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0^2})^2 + z_0^2])$$

$$(\frac{1}{\sigma_0}\sum_{j=1}^{m_i}(\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0}))dz_0\}$$

$$/ \quad \{\int_{-\infty}^{\infty}\Phi(\frac{c-\mu-\sigma_1 z_0}{\sigma_0})^{n_i-m_i}\exp\{-\frac{1}{2}[\sum_{j=1}^{m_i}(\frac{y_{ij}-\mu-\sigma_1 z_0}{\sigma_0^2})^2 + z_0^2]\}dz_0\}.$$

Clearly, w_1 is bounded and w_2 increases to infinity as y_{ij} goes to infinity. The influence function of $\hat{\sigma}_1^2$ is not bounded.

To have robustness, we need to modify our estimation procedure to ensure that unbounded terms g_3, v_2 and w_2 become bounded.

6.3 A Robust Procedure

Suppose that α_i and ϵ_{ij} are contaminated in the model $y_{ij} = \mu + \alpha_i + \epsilon_{ij}$. As noted above, the variance components estimates given by our method are sensitive to deviations from the assumed distribution. A robust procedure for limiting the influence of the deviation on the estimates is needed.

We propose an approach to obtaining resistant estimates by using Huber's least favourable density for location estimation and Huber's least favourable density for scale estimation as follows.

Using the same notation as for the 1-way model in Chapter 3, we could write the likelihood $L(\theta; Y_i)$ as

$$L(\theta; Y_i) = \int \prod_{j=1}^{m_i} f(y_{ij} \mid x) \prod_{j=m_i+1}^{n_i} P(y_{ij} < c \mid x) f(x) dx$$

where $x = \mu + \alpha_i$ and $x \sim N(\mu, \sigma_1^2)$.

Replacing the normal $N(\mu, \sigma_1^2)$ density with Huber's least favourable density for location estimation for the distribution of x, we obtain

$$f_x(x) = \begin{cases} \frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}} \exp(\frac{-(x-\mu)^2}{2\sigma_1^2}), & \text{if } |x| \le k_1 \\ \frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}} \exp(\frac{k_1^2}{2} - k_1|(x-\mu)/\sigma_1|), & \text{if otherwise} \end{cases}$$

Replacing the normal $N(0, \sigma_0^2)$ density with Huber's least favourable density for scale estimation for the distribution of ϵ_{ij}, we obtain

$$f_{\epsilon_{ij}}(t) = \begin{cases} \frac{1-\epsilon}{\sigma_0\sqrt{2\pi}} \exp(\frac{-t^2}{2\sigma_0^2}), & \text{if } |t| \le k \\ \frac{1-\epsilon}{\sigma_0\sqrt{2\pi}} \exp(\frac{-k^2}{2\sigma_0^2})(\frac{k}{|t|})^{(k^2)}, & \text{if } |t| > k \end{cases}$$

(Huber, 1981).

Note that $y_{ij} = x + \epsilon_{ij}$, so we have the form below for $f(y_{ij} \mid x)$ instead of the normal distribution $N(x, \sigma_0^2)$ for $f(y_{ij} \mid x)$ in Chapter 3

$$f(y_{ij} \mid x) = \begin{cases} \frac{1-\epsilon}{\sigma_0\sqrt{2\pi}} \exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2), & \text{if } x - \sigma_0 k \le y_{ij} < x + \sigma_0 k \\ h_1, & \text{if } \infty > y_{ij} > x + \sigma_0 k \text{ or } -\infty < y_{ij} < x - \sigma_0 k \end{cases}$$

where

$$h_1 = \frac{1-\epsilon}{\sigma_0\sqrt{2\pi}} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij} - x|})^{(k^2)}.$$

Further we obtain $P(y_{ij} < c \mid x)$ (if $k > 1$)

$$\begin{cases} -\frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}} \exp(\frac{-k^2}{2})(k)^{k^2}(|\frac{c-x}{\sigma_0}|)^{1-k^2}, & \text{if } c < x - \sigma_0 k \\ g_2, & \text{if } x - k\sigma_0 < c < x + \sigma_0 k \\ g_3, & \text{if } c > x + \sigma_0 k \end{cases}$$

where

$$g_2 = \frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}} \exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2} + (1-\epsilon)[\Phi(\frac{c-x}{\sigma_0}) + \Phi(k) - 1],$$

and

$$
\begin{aligned}
g_3 &= \frac{2(1-\epsilon)}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2} + (1-\epsilon)[2\Phi(k) - 1] \\
&+ \frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})k^{k^2}((|\frac{c-x}{\sigma_0}|)^{1-k^2} - |k|^{1-k^2}).
\end{aligned}
$$

The modified $\log L(\theta; Y_i)$ becomes

$$
\begin{cases}
\log q_1, & \text{if } |\frac{y_{ij}-x}{\sigma_0}| \le k \text{ and } c < x - \sigma_0 k \\
\log q_2, & \text{if } |\frac{y_{ij}-x}{\sigma_0}| \le k \text{ and } |\frac{c-x}{\sigma_0}| \le k \\
\log q_3, & \text{if } |\frac{y_{ij}-x}{\sigma_0}| \le k \text{ and } c > x + \sigma_0 k \\
\log q_4, & \text{if } |\frac{y_{ij}-x}{\sigma_0}| > k \text{ and } c < x - \sigma_0 k \\
\log q_5, & \text{if } |\frac{y_{ij}-x}{\sigma_0}| > k \text{ and } |\frac{c-x}{\sigma_0}| \le k \\
\log q_6, & \text{if } |\frac{y_{ij}-x}{\sigma_0}| > k \text{ and } c > x + \sigma_0 k
\end{cases}
$$

where

$$
\begin{aligned}
\log q_1 &= \log\{\int_{-\infty}^{-k_1} \prod_{j=1}^{m_i} \frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2) \\
&\prod_{j=m_i+1}^{n_i} \frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|\frac{c-x}{\sigma_0 k}|)^{1-k^2} \\
&\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{k_1^2}{2} - k_1|(\frac{x-\mu}{\sigma_1})|]dx\} \\
&+\{\int_{-k_1}^{k_1} \prod_{j=1}^{m_i} \frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2) \\
&\prod_{j=m_i+1}^{n_i} \frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|\frac{c-x}{\sigma_0 k}|)^{1-k^2} \\
&\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{-1}{2}(\frac{x-\mu}{\sigma_1})^2]dx\} \\
&+\{\int_{k_1}^{\infty} \prod_{j=1}^{m_i} \frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2) \\
&\prod_{j=m_i+1}^{n_i} \frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|\frac{c-x}{\sigma_0 k}|)^{1-k^2} \\
&\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{k_1^2}{2} - k_1|(\frac{x-\mu}{\sigma_1})|]dx\}
\end{aligned}
$$

$$
\begin{aligned}
\log q_2 \;=\; & \log\{\int_{-\infty}^{-k_1} \prod_{j=1}^{m_i} \frac{1-\epsilon}{\sigma_0\sqrt{2\pi}} \exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2) \\
& \prod_{j=m_i+1}^{n_i} \frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}} \exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2} + \\
& (1-\epsilon)[\Phi(\frac{c-x}{\sigma_0}) + \Phi(k) - 1] \\
& \frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}} \exp[\frac{k_1^2}{2} - k_1|(\frac{x-\mu}{\sigma_1})|]dx\} \\
& +\{\int_{-k_1}^{k_1} \prod_{j=1}^{m_i} \frac{1-\epsilon}{\sigma_0\sqrt{2\pi}} \exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2) \\
& \prod_{j=m_i+1}^{n_i} \frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}} \exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2} + \\
& (1-\epsilon)[\Phi(\frac{c-x}{\sigma_0}) + \Phi(k) - 1] \\
& \frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}} \exp[\frac{-1}{2}(\frac{x-\mu}{\sigma_1})^2]dx\} \\
& +\{\int_{k_1}^{\infty} \prod_{j=1}^{m_i} \frac{1-\epsilon}{\sigma_0\sqrt{2\pi}} \exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2) \\
& \prod_{j=m_i+1}^{n_i} \frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}} \exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2} + \\
& (1-\epsilon)[\Phi(\frac{c-x}{\sigma_0}) + \Phi(k) - 1] \\
& \frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}} \exp[\frac{k_1^2}{2} - k_1|(\frac{x-\mu}{\sigma_1})|]dx\}
\end{aligned}
$$

$$
\begin{aligned}
\log q_3 \;=\; & \log\{\int_{-\infty}^{-k_1} \prod_{j=1}^{m_i} \frac{1-\epsilon}{\sigma_0\sqrt{2\pi}} \exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2) \\
& \prod_{j=m_i+1}^{n_i} \frac{2(1-\epsilon)}{(1-k^2)\sqrt{2\pi}} \exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2} + (1-\epsilon)[2\Phi(k)-1] \\
& + \frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}} \exp(\frac{-k^2}{2})(k)^{k^2}((|(c-x)/\sigma_0|)^{1-k^2} - (|k|)^{1-k^2}) \\
& \frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}} \exp[\frac{k_1^2}{2} - k_1|(\frac{x-\mu}{\sigma_1})|]dx\}
\end{aligned}
$$

$$+\{\int_{-k_1}^{k_1}\prod_{j=1}^{m_i}\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2)$$
$$\prod_{j=m_i+1}^{n_i}\frac{2(1-\epsilon)}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2}+(1-\epsilon)[2\Phi(k)-1]$$
$$+\frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}((|(c-x)/\sigma_0|)^{1-k^2}-(|k|)^{1-k^2})$$
$$\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{-1}{2}(\frac{x-\mu}{\sigma_1})^2]dx\}$$
$$+\{\int_{k_1}^{\infty}\prod_{j=1}^{m_i}\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2)$$
$$\prod_{j=m_i+1}^{n_i}\frac{2(1-\epsilon)}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2}+(1-\epsilon)[2\Phi(k)-1]$$
$$+\frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}((|(c-x)/\sigma_0|)^{1-k^2}-(|k|)^{1-k^2})$$
$$\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{k_1^2}{2}-k_1|(\frac{x-\mu}{\sigma_1})|]dx\}$$

$$\begin{aligned}\log q_4 \;=\;& \log\{\int_{-\infty}^{-k_1}\prod_{j=1}^{m_i}\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)}\\
&\prod_{j=m_i+1}^{n_i}\frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|\frac{c-x}{\sigma_0 k}|)^{1-k^2}\\
&\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{k_1^2}{2}-k_1|(\frac{x-\mu}{\sigma_1})|]dx\}\\
&+\{\int_{-k_1}^{k_1}\prod_{j=1}^{m_i}\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)}\\
&\prod_{j=m_i+1}^{n_i}\frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|\frac{c-x}{\sigma_0 k}|)^{1-k^2}\\
&\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{-1}{2}(\frac{x-\mu}{\sigma_1})^2]dx\}\\
&+\{\int_{k_1}^{\infty}\prod_{j=1}^{m_i}\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)}\\
&\prod_{j=m_i+1}^{n_i}\frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|\frac{c-x}{\sigma_0 k}|)^{1-k^2}\end{aligned}$$

$$\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{k_1^2}{2}-k_1|(\frac{x-\mu}{\sigma_1})|]dx\}$$

$$\begin{aligned}
\log q_5 &= \log\{\int_{-\infty}^{-k_1}\prod_{j=1}^{m_i}\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)} \\
&\prod_{j=m_i+1}^{n_i}\frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2} \\
&+(1-\epsilon)[\Phi(\frac{c-x}{\sigma_0})+\Phi(k)-1]\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{k_1^2}{2}-k_1|(\frac{x-\mu}{\sigma_1})|]dx\} \\
&+\{\int_{-k_1}^{k_1}\prod_{j=1}^{m_i}\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)} \\
&\prod_{j=m_i+1}^{n_i}\frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2} \\
&+(1-\epsilon)[\Phi(\frac{c-x}{\sigma_0})+\Phi(k)-1]\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{-1}{2}(\frac{x-\mu}{\sigma_1})^2]dx\} \\
&+\{\int_{k_1}^{\infty}\prod_{j=1}^{m_i}\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)} \\
&\prod_{j=m_i+1}^{n_i}\frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2} \\
&+(1-\epsilon)[\Phi(\frac{c-x}{\sigma_0})+\Phi(k)-1]\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{k_1^2}{2}-k_1|(\frac{x-\mu}{\sigma_1})|]dx\}
\end{aligned}$$

$$\begin{aligned}
\log q_6 &= \log\{\int_{-\infty}^{-k_1}\prod_{j=1}^{m_i}\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)} \\
&\prod_{j=m_i+1}^{n_i}\frac{2(1-\epsilon)}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2}+(1-\epsilon)[2\Phi(k)-1] \\
&+\frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}}\exp(\frac{-k^2}{2})(k)^{k^2}((|(c-x)/\sigma_0|)^{1-k^2}-(|k|)^{1-k^2}) \\
&\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{k_1^2}{2}-k_1|(\frac{x-\mu}{\sigma_1})|]dx\} \\
&+\{\int_{-k_1}^{k_1}\prod_{j=1}^{m_i}\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)}
\end{aligned}$$

$$\prod_{j=m_i+1}^{n_i} \frac{2(1-\epsilon)}{(1-k^2)\sqrt{2\pi}} \exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2} + (1-\epsilon)[2\Phi(k)-1]$$
$$+\frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}} \exp(\frac{-k^2}{2})(k)^{k^2}((|(c-x)/\sigma_0|)^{1-k^2} - (|k|)^{1-k^2})$$
$$\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}} \exp[\frac{-1}{2}(\frac{x-\mu}{\sigma_1})^2]dx\}$$
$$+\{\int_{k_1}^{\infty} \prod_{j=1}^{m_i} \frac{1-\epsilon}{\sigma_0\sqrt{2\pi}} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)}$$
$$\prod_{j=m_i+1}^{n_i} \frac{2(1-\epsilon)}{(1-k^2)\sqrt{2\pi}} \exp(\frac{-k^2}{2})(k)^{k^2}(|-k|)^{1-k^2} + (1-\epsilon)[2\Phi(k)-1]$$
$$+\frac{1-\epsilon}{(1-k^2)\sqrt{2\pi}} \exp(\frac{-k^2}{2})(k)^{k^2}((|(c-x)/\sigma_0|)^{1-k^2} - (|k|)^{1-k^2})$$
$$\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}} \exp[\frac{k_1^2}{2} - k_1|(\frac{x-\mu}{\sigma_1})|]dx\}$$

The modified log-likelihood function can be expressed $L_r = \sum_{i=1}^{4} \log L(\theta; Y_i)$. The estimates for $\theta = (\sigma_0^2, \sigma_1^2, \mu)$ can be found by maximizing the above function L_r.

To check the influence functions of the estimates, we need to calculate $\psi_{\sigma_0^2}$, $\psi_{\sigma_1^2}$ and ψ_μ. Since $P(y_{ij} < c \mid x)$ and $f(x)$ are independent of y_{ij}, we can write $\log L_r(\theta; Y_i)$ as

$$\log L(\theta; Y_i) = \begin{cases} \log P_1, & \text{if } x - \sigma_0 k \le y_{ij} < x + \sigma_0 k \\ \log P_2, & \text{otherwise} \end{cases}$$

where

$$\begin{aligned} P_1 &= \{\int_{-\infty}^{-k_1} \prod_{j=1}^{m_i} \exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2) G(\theta, x)dx\} \\ &+\{\int_{-k_1}^{k_1} \prod_{j=1}^{m_i} \exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2) G(\theta, x)dx\} \\ &+\{\int_{k_1}^{\infty} \prod_{j=1}^{m_i} \exp(\frac{-1}{2}(\frac{y_{ij}-x}{\sigma_0})^2) G(\theta, x)dx\}, \end{aligned}$$

$$P_2 = \{\int_{-\infty}^{-k_1} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)} G(\theta, x)dx\}$$

$$+\{\int_{-k_1}^{k_1}\prod_{j=1}^{m_i}\exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)}G(\theta,x)dx\}$$
$$+\{\int_{k_1}^{\infty}\prod_{j=1}^{m_i}\exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)}G(\theta,x)dx\},$$

and $G(\theta,x)=(\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}})^{(m_i)}\prod_{j=m_i+1}^{n_i}P(y_{ij}<c\mid x)f(x)$. We can rewrite $G(\theta,x)$ as

$$G(\theta,x)=\begin{cases} G_1, & \text{if } c<x-\sigma_0 k\\ G_2, & \text{if } |\frac{c-x}{\sigma_0}|\le k\\ G_3, & \text{if } c>x+\sigma_0 k\end{cases}$$

where

$$\begin{aligned} G_1 &= (\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}})^{m_i}\prod_{j=m_i+1}^{n_i}\frac{1-\epsilon}{(1-k^2)\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\sigma_0 k)^{k^2}(\frac{c-x}{\sigma_0 k})^{1-k^2}\\ &\quad \frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{k_1^2}{2}-k_1|(\frac{x-\mu}{\sigma_1})|] \end{aligned}$$

$$\begin{aligned} G_2 &= (\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}})^{m_i}\prod_{j=m_i+1}^{n_i}\frac{2(1-\epsilon)}{(1-k^2)\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\sigma_0 k)^{k^2}(-k)^{1-k^2}\\ &\quad +(1-\epsilon)[2\Phi(k)-1]+\frac{1-\epsilon}{(1-k^2)\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\sigma_0 k)^{k^2}\\ &\quad ((c-x)^{1-k^2}-(\sigma_0 k)^{1-k^2})\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{k_1^2}{2}-k_1|(\frac{x-\mu}{\sigma_1})|], \end{aligned}$$

and

$$\begin{aligned} G_3 &= (\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}})^{m_i}\prod_{j=m_i+1}^{n_i}\frac{2(1-\epsilon)}{(1-k^2)\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\sigma_0 k)^{k^2}(-k)^{1-k^2}\\ &\quad +(1-\epsilon)[2\Phi(k)-1]+\frac{1-\epsilon}{(1-k^2)\sigma_0\sqrt{2\pi}}\exp(\frac{-k^2}{2})(\sigma_0 k)^{k^2}\\ &\quad ((c-x)^{1-k^2}-(\sigma_0 k)^{1-k^2})\frac{1-\epsilon_1}{\sigma_1\sqrt{2\pi}}\exp[\frac{k_1^2}{2}-k_1|(\frac{x-\mu}{\sigma_1})|] \end{aligned}$$

When y_{ij} and x are large, the derivative of $\log L_r(\theta;Y_i)$ with respect to σ_0^2 will be

$$\frac{\partial \log L_r(\theta, Y_i)}{\partial \sigma_0^2} =$$
$$\{\int_{-\infty}^{-k_1} (d \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij} - x|})^{(k^2)} / d\sigma_0^2) G(\theta, x) dx / L(\theta, Y_i)\}$$
$$+\{\int_{-\infty}^{-k_1} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij} - x|})^{(k^2)} (dG(\theta, x)/d\sigma_0^2) dx / L(\theta, Y_i)\}$$
$$= u_1 + u_2$$

or

$$\frac{\partial \log L_r(\theta, Y_i)}{\partial \sigma_0^2} =$$
$$\{\int_{k_1}^{\infty} (d \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij} - x|})^{(k^2)} / d\sigma_0^2) G(\theta, x) dx / L(\theta, Y_i)\}$$
$$+\{\int_{k_1}^{\infty} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij} - x|})^{(k^2)} (dG(\theta, x)/d\sigma_0^2) dx / L(\theta, Y_i)\}$$
$$= u_3 + u_4$$

where

$$u_1 = \{\int_{-\infty}^{-k_1} \frac{m_i k^2}{\sigma_0} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij} - x|})^{(k^2)} G(\theta, x) dx\} / L(\theta, Y_i) = \frac{m_i k^2}{\sigma_0}$$

and

$$u_2 = \{\int_{-\infty}^{-k_1} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij} - x|})^{(k^2)} . (dG(\theta, x)/d\sigma_0^2) dx\} / L(\theta, Y_i)$$

$$u_3 = \{\int_{k_1}^{\infty} \frac{m_i k^2}{\sigma_0} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij} - x|})^{(k^2)} G(\theta, x) dx\} / L(\theta, Y_i) = \frac{m_i k^2}{\sigma_0}$$

and

$$u_4 = \{\int_{k_1}^{\infty} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij} - x|})^{(k^2)} . (dG(\theta, x)/d\sigma_0^2) dx\} / L(\theta, Y_i)$$

u_1 and u_3 are independent of y_{ij} and x. Let us look at u_2 and u_4. We can see that $G(\theta, x)$ is not zero. Say there is $m_1 \neq 0$ $m_1 \leq G(\theta, x)$. $dG(\theta, x)/d\sigma_0^2$ involves $d(\frac{1-\epsilon}{\sigma_0\sqrt{2\pi}})/d\sigma_0^2$, $d\Phi(\frac{c-x}{\sigma_0})/d\sigma_0^2$, and $d(\frac{c-x}{\sigma_0 k})^{(1-k^2)})/d\sigma_0^2$. They are all bounded. Therefore $dG(\theta, x)/d\sigma_0^2$ is bounded. Say $dG(\theta, x)/d\sigma_0^2 \leq M$. Thus

$$0 \leq u_2 \leq \frac{M \int_{-\infty}^{\mu - k_1\sigma_1} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij} - x|})^{(k^2)} dx}{m_1 \int_{-\infty}^{\mu - k_1\sigma_1} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij} - x|})^{(k^2)} dx} = \frac{M}{m_1}$$

and

$$0 \leq u_4 \leq \frac{M \int_{\mu+k_1\sigma_1}^{\infty} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)} dx}{m_1 \int_{\mu+k_1\sigma_1}^{\infty} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)} dx} = \frac{M}{m_1}.$$

Since u_1, u_2, u_3, and u_4 are all bounded, $\psi_{\sigma_0^2}$ is bounded when y_{ij} and x increase.

To get $\psi_{\sigma_1^2}$ and ψ_μ, we consider

$$\frac{\partial \log L(\theta, Y_i)}{\partial \sigma_1^2} = \frac{1}{L(\theta, Y_i)} \frac{\partial L(\theta, Y_i)}{\partial x} \frac{\partial x}{\partial \sigma_1^2}$$

and

$$\frac{\partial \log L(\theta, Y_i)}{\partial \mu} = \frac{1}{L(\theta, Y_i)} \frac{\partial L(\theta, Y_i)}{\partial x} \frac{\partial x}{\partial \mu}.$$

It is easy to see that

$$\frac{\partial L(\theta, Y_i)}{\partial x} = \frac{\partial \int_{-\infty}^{-k_1} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)} G(\theta, x) dx}{\partial x} = 0$$

and

$$\frac{\partial L(\theta, Y_i)}{\partial x} = \frac{\partial \int_{k_1}^{\infty} \prod_{j=1}^{m_i} \exp(\frac{-k^2}{2})(\frac{\sigma_0 k}{|y_{ij}-x|})^{(k^2)} G(\theta, x) dx}{\partial x} = 0.$$

Note that $\frac{1}{L(\theta, Y_i)}$ is bounded, $\frac{\partial x}{\partial \sigma_1^2} = z_0$, and $\frac{\partial x}{\partial \mu} = 1$. We have that $\psi_{\sigma_1^2}$ and ψ_μ are bounded.

We will have a bounded influence function as y_{ij} and x both increase if we use the modified likelihood function L_r.

6.4 Examples

We implemented a small simulation study to investigate the performance of the estimators presented above.

Example 6.3. Using the data in Example 6.1, we give the estimates by our robust procedure (MIS_{rob}, $k = 2.46$, and $k_1 = 1.399$). Table 6.3 illustrates the results, compared with those three methods in Example 6.1

It can be seen that MIS_{rob} gives better estimates with complete data.

Table 6.3: $\sigma_0^2 = 1.0$, $\sigma_1^2 = 1.0$ *and* $\mu = 0.0$ *with complete data*

	σ_0^2	σ_1^2	μ
ANOVA	4.7694	2.5854	-0.0388
MLE	4.7973	1.9174	-0.0348
MIS	4.7173	2.3957	-0.0206
MIS_{rob}	1.3258	0.7779	-0.0252

Example 6.4. Using the same largest 22 observations as in Example 6.2, we list the results by four methods in Table (6.4) ($k = 2.46$, and $k_1 = 1.399$ for MIS_{rob})

Table 6.4: $\sigma_0^2 = 1.0$, $\sigma_1^2 = 1.0$ *and* $\mu = 0.0$ *with data which are greater than 0.0*

	σ_0^2	σ_1^2	μ
ANOVA	3.2895	0.1251	1.3242
MLE	3.0862	0.1702	1.2478
MIS	3.0441	0.4506	0.0100
MIS_{rob}	0.9968	0.4504	0.0100

It appears from the table 6.4 that the MIS_{rob} improves the estimates.

Example 6.5. 11 groups of data are generated using the same model and the same parameters as in Example 6.1. The ϵ_{ij} have distribution

$$0.95N(0, \sigma_0^2) + 0.05N(0, 50\sigma_0^2).$$

The observations which are less then 0.0 are removed as missing data. Table 6.5 gives the numerical results of four methods ($k = 1.81$, and $k_1 = 1.399$)

The mean squared error (MSE) for each estimates by MIS and MIS_{rob} methods are summarized as follows

- $MSE_{MIS}(\sigma_0^2) = 13.866$, $MSE_{MIS_{rob}}(\sigma_0^2) = 0.2398$,

- $MSE_{MIS}(\sigma_1^2) = 0.4986$, $MSE_{MIS_{rob}}(\sigma_1^2) = 0.4076$, item $MSE_{MIS}(\mu) = 0.0808$, $MSE_{MIS_{rob}}(\mu) = 0.0309$.

We can see that MIS_{rob} gives better estimates when contamination is present. The numerical results are consistent with the influence function study.

Table 6.5: $\sigma_0^2 = 1.0$, $\sigma_1^2 = 1.0$, $\mu = 0.0$ *with data which are great than* 0.0

	σ_0^2	σ_1^2	μ		σ_0^2	σ_1^2	μ
ANOVA	3.8945	1.0607	1.5604	ANOVA	1.4187	1.2109	1.8213
MLE	4.1514	0.3790	1.5725	MLE	1.4873	8.3308	2.8558
MIS	6.1759	1.0069	0.0100	MIS	3.9628	0.3600	0.0100
MIS_{rob}	0.8227	0.5989	0.0100	MIS_{rob}	1.8779	1.2249	0.0100
ANOVA	0.4832	11.7475	1.8647	ANOVA	0.6309	0.2109	1.2636
MLE	0.4841	20.7956	3.0528	MLE	0.6139	0.1117	1.1603
MIS	1.2876	0.9155	0.0100	MIS	0.9983	0.3745	0.3319
MIS_{rob}	1.2863	0.9084	0.0100	MIS_{rob}	1.0190	0.3711	0.3117
ANOVA	8.3405	0.6433	1.6960	ANOVA	0.3116	2.2732	1.4408
MLE	8.4691	0.0000	1.6939	MLE	0.3171	2.7455	1.8468
MIS	11.3359	0.0232	0.0100	MIS	0.8124	0.2082	0.0100
MIS_{rob}	2.0091	0.0000	0.0100	MIS_{rob}	0.8423	0.6270	0.0100
ANOVA	1.1426	0.0611	1.0758	ANOVA	1.1653	9.0148	1.6228
MLE	1.0655	0.0623	1.0932	MLE	1.1736	24.0344	3.5973
MIS	2.4233	0.1625	0.0100	MIS	3.1263	0.1399	0.5402
MIS_{rob}	0.9419	0.0000	0.0100	MIS_{rob}	0.7785	0.7187	0.0851
ANOVA	0.7296	0.0023	0.8799	ANOVA	1.7474	0.7022	1.9607
MLE	0.6805	0.0101	0.8716	MLE	1.6772	0.4799	1.7489
MIS	1.3110	0.0236	0.0100	MIS	2.8349	1.2763	0.0253
MIS_{rob}	0.6451	0.0236	0.0100	MIS_{rob}	1.6168	1.2751	0.0100
ANOVA	0.8399	0.0798	1.1710				
MLE	0.8139	0.0799	1.1418				
MIS	1.1246	0.2096	0.6969				
MIS_{rob}	1.1045	0.2094	0.4847				

Chapter 7

Estimation of Random Effects

Chapter 1 has introduced the general linear mixed model

$$Y = X\beta + Z\gamma + \epsilon,$$

where Y is the vector of observations, X and Z are known design matrices, β is a vector of fixed effects, γ is a vector of random effects, assumed to be distributed as $N(\mu, \Sigma)$, and ϵ is a vector of error terms, distributed as $N(0, \sigma_0^2 I)$, and $cov(\gamma, \epsilon) = 0$.

This chapter will discuss a practical problem associated with the model – prediction of γ (or estimation of random effects).

7.1 Introduction

Consider measuring intelligence in humans. Each of us has some level of intelligence. It can never be measured exactly. As a substitute, we have test scores which are used for putting a value to an individual's IQ. Psychologists use test scores to predict a person's intelligence. Here y is the vector of test scores and γ is the unknowable true value of a person's intelligence. If $\hat{\gamma}$ denotes the estimate of γ, $\hat{\gamma}$ will be the prediction of a person's intelligence. There are many situations similar to that of the people's IQ where we want to quantify the realization of an unobservable random variable.

In particular in our examples, the unobservable random variable is the genetic merit in our fish breeding set-up. Each individual fish has its own genetic merit which can not be measured. We have the length (or weight) of his (or her) offspring. We want to predict an individual's genetic merit by using the observable length (or weight).

A statement of the general problem is easy. Suppose Y and γ are jointly distributed vectors of random variables, with those in Y being observable but those in γ not being observable. The problem is to estimate γ from observed value of Y. Usually Y contains more elements than γ.

7.2 Estimation with Complete Data

Three methods of prediction are of interest:

- Best prediction (BP);
- best linear prediction (BLP);
- best linear unbiased prediction (BLUP).

The best predictor of γ is the conditional mean of γ given Y

$$BP(\gamma) = E(\gamma \mid Y).$$

If (Y, γ) is multivariate normal

$$\begin{pmatrix} Y \\ \gamma \end{pmatrix} \sim N(\begin{pmatrix} \mu_Y \\ \mu_\gamma \end{pmatrix}, \begin{pmatrix} V & C \\ C' & \Sigma \end{pmatrix})$$

with $C = \Sigma Z'$,

$$BP(\gamma) = E(\gamma \mid Y) = \mu_\gamma + C'V^{-1}(Y - \mu_Y).$$

We can see that the predictor cannot be estimated without having values for μ_γ, μ_Y, C, and V. Thus the best predictor is available when we know all the parameters of the joint distribution of Y and γ.

For best linear prediction, we assume predictor is linear in Y, of the form

$$\hat{\gamma} = a + BY$$

for some vector a and matrix B. Minimizing

$$\int\int (\hat{\gamma} - \gamma)' A (\hat{\gamma} - \gamma) f(\gamma, Y) dY d\gamma$$

leads (without any assumption of normality) to

$$BLP(\gamma) = \mu_\gamma + CV^{-1}(Y - \mu_Y).$$

We still need knowledge of μ_γ, μ_Y, C, and V but without assuming normality. $BLP(\gamma)$ is identical to $BP(\gamma)$ under normality. Thus the best linear predictor is available when we know all the parameters.

The BLUP (Best Linear Unbiased Prediction) of γ is a statistical methodology that has been used extensively. Harville (1976) derived this estimate by extending the Gauss-Markov theorem to cover random effects

$$BLUP(\gamma) = E(\gamma \mid Y, \hat{\beta}, V) = Var(\gamma) Z' V^{-1} (y - X\hat{\beta})$$

where

$$\hat{\beta} = (X'V^{-1}X)^{-1} X'V^{-1}Y$$

(Robinson, 1991). It can be see that BLUP is available when we know V (V is replaced by an estimate in practice).

7.3 Estimation with Missing Data

When Y is observed as $(y_1, \ldots, y_m, y_{m+1} < c, \ldots, y_n < c)$, we could not apply the formulas which are given in the last section to estimate γ because of not having a complete data vector Y. It is of interest to develop a method to estimate γ with $(y_1, \ldots, y_m, y_{m+1} < c, \ldots, y_n < c)$. To do this use a Bayesian approach as follows.

- The prior density of γ

  We regard γ as a parameter which has a prior distribution $N(0, \Sigma)$. The prior density for γ is

$$\pi(\gamma) = (2\pi)^{-q/2}(\det \Sigma)^{-1/2} \exp \frac{-1}{2}(\gamma)'\Sigma^{-1}(\gamma).$$

- The conditional density of $(Y \mid \beta, \gamma)$

  Model 1.1 can easily be rewritten as

$$y_i = \sum_{j=1}^{p} x_{ij}\beta_j + \sum_{k=1}^{q} z_{ik}\gamma_k + \epsilon_i$$

  for $i = 1, \ldots, n$.

  Under normality, the distribution of $(y_i \mid \beta, \gamma)$ will be

$$N(\sum_{j=1}^{p} x_{ij}\beta_j + \sum_{k=1}^{q} z_{ik}\gamma_k, \sigma_0^2).$$

  Since $y_1, \ldots, y_n$ are independent when γ is given, the conditional density of $(Y \mid \beta, \gamma)$ can be written as

$$\begin{aligned} f(Y \mid \beta, \gamma) &= \prod_{i=1}^{m} f(y_i \mid \gamma) \prod_{i=m+1}^{n} P(y_i < c \mid \gamma) \\ &= \prod_{i=1}^{m}(2\pi\sigma_0^2)^{-1/2} \exp \frac{-1}{2\sigma_0^2}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \sum_{k=1}^{q} z_{ik}\gamma_k)^2 \\ &\quad \prod_{i=m+1}^{n} \Phi(\frac{c - \sum_{j=1}^{p} x_{ij}\beta_j - \sum_{k=1}^{q} z_{ik}\gamma_k}{\sigma_0}). \end{aligned}$$

- The posterior density of γ

Therefore the posterior density of γ is

$$\pi(\gamma \mid Y) \propto \frac{f(Y \mid \beta, \gamma)\pi(\gamma)}{f(Y)}.$$

We get

$$\begin{aligned}\pi(\gamma \mid Y) &= \{\prod_{i=1}^{m}(2\pi\sigma_0^2)^{-1/2}\exp\frac{-1}{2\sigma_0^2}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j - \sum_{k=1}^{q} z_{ik}\gamma_k)^2 \\ &\prod_{i=m+1}^{n}\Phi(\frac{c - \sum_{j=1}^{p} x_{ij}\beta_j - \sum_{k=1}^{q} z_{ik}\gamma_k}{\sigma_0})(2\pi)^{-q/2}(\det \Sigma)^{-1/2} \\ &\exp(\frac{-1}{2}\gamma'\Sigma^{-1}\gamma)\}/f(Y). \qquad (7.1)\end{aligned}$$

where $f(Y)$ is the density function of $(y_1, \ldots, y_m, y_{m+1} < c, \ldots, y_n < c)$ (Robinson, 1991). For the one-way model, $f(Y)$ is function (3.3) as we have derived in Chapter 3, and $f(Y)$ will be function (4.5) of Chapter 4 for two way nested model.

Estimation of γ can be accomplished by maxmizing the posterior $\pi(\gamma \mid Y)$ where Y is $(y_1, \ldots, y_m, y_{m+1} < c, \ldots, y_n < c)$.

If β, σ_0^2, σ_1^2, and σ_2^2 are known, we could estimate γ by $max_\gamma \pi(\gamma \mid Y)$. The computation is straightforward since we do not really need to consider the denominator of (7.1). It is just a function of Y and so, given Y, is effectively a constant. For numerical calculations, one of the optimization routines in NAG can be used to get the results. With σ_0^2, σ_1^2, and σ_2^2 unknown, a common practice is to replace them by the estimates $\hat{\sigma_0^2}$, $\hat{\sigma_1^2}$, and $\hat{\sigma_2^2}$ in expressions in $\pi(\gamma \mid Y)$. The estimates of σ_0^2, σ_1^2, and σ_2^2 which we have developed in the previous chapters would be reasonable estimates here.

If we let β, σ_0^2, σ_1^2, and σ_2^2 be unknown parameters in $\pi(\gamma \mid Y)$, the calculation of

$$max_{\gamma, \beta, \sigma_0^2, \sigma_1^2, \sigma_2^2}\pi(\gamma \mid Y)$$

can be extremely difficult to carry out due to the high-dimension maximization.

7.4 Example

To illustrate (7.1) we use the 1-way model of Chapter 3. It has model equation $y_{ij} = \mu + \alpha_i + \epsilon_{ij}$. Suppose $Y = (y_{11}, y_{12}, y_{21}, y_{22} < c)$, $\mu = 0$, and $\sigma_0^2 = \sigma_1^2 = 1$, the conditional density becomes

$$\begin{aligned} f(Y \mid \gamma) &= \frac{1}{(2\pi)^{(3/2)}} \exp \frac{-1}{2}[(y_{11} - \alpha_1)^2 + (y_{12} - \alpha_1)^2 + (y_{21} - \alpha_2)^2] \\ &\quad \Phi(c - \alpha_2) \end{aligned}$$

where $\gamma = (\alpha_1, \alpha_2)'$.

Note also that the prior of γ is

$$\pi(\gamma) = \frac{1}{2\pi} \exp \frac{-1}{2}(\alpha_1^2 + \alpha_2^2).$$

Hence the posterior density (7.1) is proportional to

$$\begin{aligned} \pi(\gamma \mid Y) &\propto \frac{1}{(2\pi)^{5/2}} \exp \frac{-1}{2}[(y_{11} - \alpha_1)^2 + (y_{12} - \alpha_1)^2 + (y_{21} - \alpha_2)^2 + \alpha_1^2 + \alpha_2^2] \\ &\quad \Phi(c - \alpha_2). \end{aligned} \tag{7.2}$$

Taking log of the right-hand side of (7.2) and ignoring terms that are not functions of γ, we obtain

$$l = \frac{-1}{2}[(y_{11} - \alpha_1)^2 + (y_{12} - \alpha_1)^2 + (y_{21} - \alpha_2)^2 + \alpha_1^2 + \alpha_2^2] + \log(\Phi(c - \alpha_2)).$$

Differentiating this expression with respect to α_1 and α_2 will yield

$$\frac{\partial l}{\partial \alpha_1} = (y_{11} - \alpha_1) + (y_{12} - \alpha_1) - \alpha_1,$$

$$\frac{\partial l}{\partial \alpha_2} = (y_{21} - \alpha_2) - \alpha_2 - \frac{\phi(c - \alpha_2)}{\Phi(c - \alpha_2)}.$$

Equating these two expressions to zero gives

$$y_{11} + y_{12} - 3\alpha_1 = 0$$

and

$$y_{21} - 2\alpha_2 - \frac{\phi(c - \alpha_2)}{\Phi(c - \alpha_2)} = 0.$$

$\hat{\alpha}_1 = (y_{11} + y_{12})/3$ and the solution of $y_{21} - 2\alpha_2 - \frac{\phi(c-\alpha_2)}{\Phi(c-\alpha_2)} = 0$ are the estimates of the random effects $\gamma = (\alpha_1, \alpha_2)'$.

Chapter 8

Concluding Remarks

In this thesis, point estimates and approximate confidence intervals of variance components with a high proportion of missing data have been derived both for one-way and two-way nested models. Despite recent advances in the analysis of data with missing values, very little work has been done on variance components estimation with missing data. The major difficulty of this subject is that the observations are not independent. We can not write the full likelihood as we usually do in survival analysis

$$lik = \prod_{obs} f(t;\phi) \prod_{mis} F(c;\phi).$$

It will be also hard to apply the EM algorithm to this subject because $P(\theta \mid y_{obs}, y_{mis})$ can not be written as linear in the unobserved data y_{mis} (Little and Rubin pointed out that estimates can be severely biased when the EM approach is applied in general, 1983). Chapter 3 (section 3.5) gives more details about the difficulty of using the EM algorithm to estimate variance components with incomplete Y. Therefore, our results are particularly useful.

In our model-based procedure, a full likelihood function is defined, in which the missing information has been taken into account. This likelihood function is transformed into a computable function which is maximized to get the estimates. Our method is applied to simulated data and aquacultural data. The results obtained are significantly and uniformly more accurate than those obtained by any of the standard methods. Different issues concerning the method (such as the existence, uniqueness, confidence intervals, robust procedure, and random effects estimation) have been studied in the thesis.

Future work will continue in several directions. Knowledge, or absence of knowledge, of the mechanisms that led to certain values being missing from an observed distribution is a key element in choosing an appropriate analysis and in interpreting the results. The mechanism that led to missing data in the selective genotyping method which we describe here is a form where the threshold is fixed. In some situations, the threshold may not be known exactly. Probabilistic thresholds may be a characteristic of many populationsi, where the probability that data is observed increases as the value of data increases. This situation will arise, for example, when grading and scoring procedures are imperfect. Thus most of the observed data are large ones. We are currently working on this.

An assumption is being made in our procedure that the family size (n_{ij}) is known. If the family sizes are unknown, the problem will be much more difficult. If we treated all n_{ij} as unknown parameters, there will be $\sum_{ij} n_{ij} + 4$ parameters for the two-way model. The computation of the constrained nonlinear optimization will be very difficult due to the large number of parameters. An alternative is to estimate n_{ij} first, thereby decreasing the number of parameters being optimized. How this will effect the estimates of variance components has to be investigated.

Future investigation includes extending our method to different designs and getting robust procedure for different designs. In addition the global search of the parameter space to solve our optimization problem is not very effective.

Appendix A

Source Code of Programmes Used

Three programmes were used for all the compuation in this thesis. The source code of these programmes are in C++. Several source files may involve for a programme. These programmes are listed below with the name(s) of source file involved:

- one-way (ow.cpp),
- one-way robust (owrub.cpp),
- two-way nested (mutw.hpp, mutw.cpp, and tw.cpp).

These source codes were written for the Borland C++ (3.0/3.1) compiler under PC/DOS.

A.1 One-Way

```
#include <conio.h>
#include <math.h>
#include <graphics.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define NUMB_OF_GROUP        5
#define NUMB_PER_GROUP       8
```

```
    typedef signed char   Boolean;
    typedef unsigned char UCHAR;

#ifdef EIGHTY_BITS
    #define HUGE        4900
    #define H_VAL       1.0E+4900
    #define EP          1.0E-4900
    #define TINY        -4900.0
    #define EXP(x)      expl( x )
    #define LOG(x)      logl( x )
    typedef long double MY_TYPE;
#else
    #define HUGE        300
    #define H_VAL       1.0E+300
    #define EP          1.0E-300
    #define TINY        -300.0
    #define EXP(x)      exp( x )
    #define LOG(x)      log( x )
    typedef double      MY_TYPE;
#endif

    #define NUMB_OF_NODE          10
    #define NUMB_SEARCH_STEP       5

    const  UCHAR      OW_ANOVA = 0x01;
    const  UCHAR      OW_MLE   = 0x02;
    const  UCHAR      OW_MM    = 0x04;


    void fatal_err( char *msg )
    {
    printf( "Error: %s!\n", msg );
    exit(1);
    }/* end of fatal_err(...) */


    float  obs[ 40 ] = {
    1.632,  1.974,  2.411,  1.370,  1.381,  2.419,  2.841,  3.341,
    0.691,  0.777,  2.202,  2.206,  0.983,
    0.883,  1.279,  0.702,
    0.232,  1.292,  3.529,  1.644,  1.545,  0.679,  1.444,
    1.470,  0.832
    };

    short  data_struct[ 5 ] = {8, 5, 3, 7, 2};

    MY_TYPE Erf( MY_TYPE x )
    {
    static MY_TYPE a[] = { 0.0705230784, 0.0422820123, 0.0092705272,
                           0.0001520143, 0.0002765672, 0.0000430638 };
    MY_TYPE y = 1.0, xx = x;
    short i;

    for ( i=0;i<6;i++ ) {
         y += a[i]*xx;
         xx *= x;
         }
    return pow( y, -16.0 );
    }/* end of MY_TYPE Erf(...) for Phi(...) */
```

```
MY_TYPE Phi( MY_TYPE u )
{
if ( u>=15.0 )
    return 1.0;
if ( u<=-15.0 )
    return 0.0;

if ( u>0.0 )
    return 0.5*( 2.0 - Erf( u*0.7071067812 ) );
return 0.5*Erf( -u*0.7071067812 );
}                               /* Phi(...) */


class ONEWAY
{
public:
    ONEWAY( float *_Obs, short *_DataStruct );
    ~ONEWAY( void );
    virtual void SetData( float *_Obs, short *_DataStruct );
    float   *GetResult( UCHAR opt=OW_ANOVA );
    void    ShowResult( UCHAR opt=OW_ANOVA|OW_MLE );
protected:
    virtual void  DOIT( void );
    void    anova( void );
    float   mle( float *xx );
    float   Optimize( float *lx, float *dx );
    char    *FileName, IsDONE;
    short   NoOfData, *DatainGrp;
    float   *Data;
    float   x[ 10 ];
    FILE    *in;
};


ONEWAY::ONEWAY( float *_Obs, short *_ObsinGrp )
{
DatainGrp = NULL;
Data = NULL;
SetData( _Obs, _ObsinGrp );
FileName = NULL;
IsDONE = 0;
}//End of ONEWAY::ONEWAY(...)


ONEWAY::~ONEWAY( void )
{
delete [] Data;
delete [] DatainGrp;
if ( FileName )
    delete [] FileName;
else
    fclose( in );
}//End of ONEWAY::~ONEWAY()


void ONEWAY::SetData( float *_Obs, short *_ObsinGrp )
{
if ( DatainGrp )
    delete [] DatainGrp;
DatainGrp = new short[ NUMB_OF_GROUP ];
```

```
if (!DatainGrp)
    fatal_err( "No Memory" );

NoOfData = 0;
for ( short i=0;i<NUMB_OF_GROUP;i++ ) {
    DatainGrp[i] = _ObsinGrp[ i ];
    NoOfData += DatainGrp[ i ];
    }
if ( Data )
    delete [] Data;

Data = new float[ NoOfData ];
if (!Data)
    fatal_err( "No Memory" );

for ( i=0;i<NoOfData;i++ )
    Data[ i ] = _Obs[ i ];
IsDONE = 0;
}//End of ONEWAY::SetData(...)


float *ONEWAY::GetResult( UCHAR opt )
{
if ( !IsDONE )
     DOIT();
if ( opt&OW_ANOVA )
    return x;
if ( opt&OW_MLE )
    return &x[3];
else
    return &x[6];
}//End of ONEWAY::GetResult(...)


void ONEWAY::ShowResult( UCHAR opt )
{
if ( !IsDONE )
    DOIT();
printf("ANOVA sigma_o %f  sigma_1 %f  mu %f\n", x[0], x[1], x[2] );
printf("MLE   sigma_o %f  sigma_1 %f  mu %f\n", x[3], x[4], x[5] );
if ( opt&OW_MM )
    printf("MM    sigma_o %f  sigma_1 %f  mu %f\n", x[6], x[7],
x[8] );
}//End of ONEWAY::ShowResult(...)


void ONEWAY::DOIT( void )
{
short i;
float lx[4], ux[4], dx[4], oldoptf, optf, c;

IsDONE = 1;
anova();
lx[0] = lx[1] = lx[2] = 0.01;
for ( i=0;i<3;i++ ) {
    ux[i] = x[ i ]*2.0;
    dx[i] = ( ux[i] - lx[i] )/NUMB_SEARCH_STEP;
    }

oldoptf = 1.0; optf = 0.0;
```

```
while ( fabs( optf - oldoptf )>0.0001 ) {
    oldoptf = optf;
    optf = Optimize( lx, dx );
    for ( i=0;i<3;i++ ) {
        c = (ux[i] - lx[i])/NUMB_SEARCH_STEP;
        lx[i] = x[ 3+i ] - c;
        if ( lx[i]<0.01 )
            lx[i] = 0.01;
        ux[i] = x[ 3+i ] + c;
        dx[i] = (ux[i] - lx[i])/NUMB_SEARCH_STEP;
        }
    }
}//End of ONEWAY::DOIT()


void ONEWAY::anova( void )
{
short i, j, k;
float s1, s2, s3, a;
k = 0;
s1 = s2 = s3 = x[0] = x[1] = 0.0;
for ( i=0;i<NUMB_OF_GROUP;i++ ) {
    x[0] += DatainGrp[i];
    x[1] += DatainGrp[i]*DatainGrp[i];
    a = 0.0;
    for ( j=0;j<DatainGrp[i];j++ ) {
        s1 += obs[ k+j ]*obs[ k+j ];
        a += obs[ k+j ];
        }
    k += DatainGrp[i];
    s2 += a*a/DatainGrp[i];
    s3 += a;
    }
x[5] = x[2] = s3/x[0];
s3 *= s3;
s3 /= x[0];
a =( x[0]*(NUMB_OF_GROUP-1) )/( x[0]*x[0] - x[1] );
x[0] = ( s1 - s2 )/( x[0] - NUMB_OF_GROUP );
x[1] = a*( ( s2 - s3 )/( NUMB_OF_GROUP - 1 ) - x[0] );
}//End of ONEWAY::anova()


float ONEWAY::mle( float *xx )
{
short  j, i, l;
float  a, b, c, d, e, f;
a = 1.0/xx[0];
f = l = 0;
for ( i=0;i<NUMB_OF_GROUP;i++ ) {
    f += log( pow( xx[0], DatainGrp[i]-1 )*
( xx[0] + DatainGrp[i]*xx[1] ) );
    b = xx[1]/( xx[0]*xx[0] + DatainGrp[i]*xx[0]*xx[1] );
    d = e = 0.0;
    for ( j=0;j<DatainGrp[i];j++ ) {
        c = Data[ l+j ] - xx[ 2 ];
        d += c*c;
        e += c;
        }
    f += a*d - b*e*e;
    l += DatainGrp[i];
```

```
        }
    return f;
    }//End of ONEWAY::mle()


    float  ONEWAY::Optimize( float  lx, float *dx )
    {
    float  xx[ 4 ], f, optf;
    short  j, i, k;

    optf = 1.0E+8;
    xx[0] = lx[0];
    for ( i=0;i<=NUMB_SEARCH_STEP;i++ ) {
/* loop for sigma_o */
        xx[1] = lx[1];
        for ( j=0;j<=NUMB_SEARCH_STEP;j++ ) {
/* loop for sigma_1 */
            xx[2] = lx[2];
            for ( k=0; k<=NUMB_SEARCH_STEP;k++) {
                f = mle( xx );
                if ( f<optf ) {
                    optf = f;
                    x[ 3 ] = xx[ 0 ];
                    x[ 4 ] = xx[ 1 ];
                    x[ 5 ] = xx[ 2 ];
                    }
                xx[2] += dx[2];
                }
            xx[1] += dx[1];
            }
        xx[0] += dx[0];
        }
    return optf;
    }//End of ONEWAY::Optimize(...)


    class ONEWAY_MM : public ONEWAY
    {
    public:
        ONEWAY_MM( float *_Obs, short *_ObsinGrp, float _TruncateV );
        ~ONEWAY_MM( void );
        void   DOIT( void );
    protected:
        float  Optimize( float *lx, float *dx );
        void   SetNodes( void );
        float  func( float *xx );
        MY_TYPE IntgrT( float *xx, short ith_grp );
        short  *missing, *GrpPtr;
        float  TruncatedValue;
        MY_TYPE gx[NUMB_OF_NODE], gw[NUMB_OF_NODE];
        MY_TYPE Node[NUMB_OF_NODE], dv[NUMB_OF_NODE];
    };


    ONEWAY_MM::ONEWAY_MM( float *_Obs, short *obs_per_grp,
float _TruncatedV ) :
        ONEWAY( _Obs, obs_per_grp )
    {
    missing = new short[ NUMB_OF_GROUP ];
```

```
GrpPtr = new short[ NUMB_OF_GROUP ];

if (!missing || !GrpPtr)
    fatal_err( "No Memory" );

for ( short i=0;i<NUMB_OF_GROUP;i++ )
    missing[i] = NUMB_PER_GROUP - obs_per_grp[i];

GrpPtr[0] = 0;
for ( i=1;i<NUMB_OF_GROUP;i++ )
    GrpPtr[i] = GrpPtr[i - 1] +  obs_per_grp[i - 1];

TruncatedValue = _TruncatedV;
SetNodes();
}//End of ONEWAY_MM::ONEWAY_MM(...)


ONEWAY_MM::~ONEWAY_MM( void )
{
delete [] GrpPtr;
delete [] missing;
}//End of ONEWAY_MM::~ONEWAY_MM()


void ONEWAY_MM::DOIT( void )
{
short i;
float lx[4], ux[4], dx[4], oldoptf, optf, c;

ONEWAY::DOIT();

lx[0] = lx[1] =  lx[2] = 0.01;
for ( i=0;i<3;i++ ) {
    ux[i] = x[ 3+i ]*2.0;
    dx[i] = ( ux[i] - lx[i] )/NUMB_SEARCH_STEP;
    }

oldoptf = 1.0; optf = 0.0;
while ( fabs( optf - oldoptf )>0.0001 ) {
    oldoptf = optf;
    optf = Optimize( lx, dx );
    for ( i=0;i<3;i++ ) {
        c = (ux[i] - lx[i])/NUMB_SEARCH_STEP;
        lx[i] = x[ 6+i ] - c;
        if ( lx[i]<0.01 )
            lx[i] = 0.01;
        ux[i] = x[ 6+i ] + c;
        dx[i] = (ux[i] - lx[i])/NUMB_SEARCH_STEP,
        }
    }
}//End of ONEWAY_MM::DOIT()


float  ONEWAY_MM::Optimize( float *lx, float *dx )
{
float  xx[ 4 ], f, optf;
short  j, i, k;

optf = 1.0E+8;
xx[2] = lx[2];
```

```
    for ( k=0;k<=NUMB_SEARCH_STEP;k++ ) {
        xx[0] = lx[0];
        for ( i=0;i<=NUMB_SEARCH_STEP;i++ ) {
/* loop for sigma_o */
            xx[1] = lx[1];
            for ( j=0;j<=NUMB_SEARCH_STEP;j++ ) {
/* loop for sigma_1 */
                f = func( xx );
                if ( f<optf ) {
                    optf = f;
                    x[ 6 ] = xx[ 0 ];
                    x[ 7 ] = xx[ 1 ];
                    x[ 8 ] = xx[ 2 ];
                    }
                xx[1] += dx[1];
                }
            xx[0] += dx[0];
            }
        xx[2] += dx[2];
        }
    return optf;
    }//End of ONEWAY_MM::Optimize(...)


    void ONEWAY_MM::SetNodes( void )
    {
    gx[0] = 0.98695326;  gw[0] = 0.033335672;
    gx[1] = 0.93253168;  gw[1] = 0.07472567;
    gx[2] = 0.83970478;  gw[2] = 0.10954318;
    gx[3] = 0.71669770;  gw[3] = 0.13463336;
    gx[4] = 0.57443717;  gw[4] = 0.14776211;
    gx[5] = 0.42556283;  gw[5] = 0.14776211;
    gx[6] = 0.28330230;  gw[6] = 0.13463336;
    gx[7] = 0.16029522;  gw[7] = 0.10954318;
    gx[8] = 0.06746832;  gw[8] = 0.07472567;
    gx[9] = 0.01304674;  gw[9] = 0.033335672;

    for ( short i=0;i<NUMB_OF_NODE;i++ )
        {
        dv[i] = ( 1.0- gx[i] ) / gx[i];
        Node[i] = LOG( gw[i] ) - 0.5*dv[i]*dv[i] - 2.0*LOG( gx[i] );
        }
    }// End of void ONEWAY_MM::SetNodes()


    float ONEWAY_MM::func( float *xx )
    {
    short i;
    float f = 0.0;
    MY_TYPE ff;

    xx[0] = sqrt( xx[0] );
    xx[1] = sqrt( xx[1] );

    for ( i=0; i<NUMB_OF_GROUP;i++ )
         {
         ff = IntgrT( xx, i );
         if (ff > EP)
```

```
            f -= LOG( ff );
        else
            f -= TINY;
        }

    xx[0] *= xx[0];
    xx[1] *= xx[1];
    return f;
    }//End of ONEWAY_MM::func(...)


    MY_TYPE ONEWAY_MM::IntgrT( float *xx, short ith_grp )
    {
    short  j, node_i;
    float * y_ij = &Data[ GrpPtr[ith_grp] ];
    MY_TYPE w, z, invxx0, phi, d, e, s;

    invxx0 = 1.0/xx[0];
    e = DatainGrp[ith_grp];
    e *= LOG( xx[0] );

    s = 0.0;
    for (node_i = 0; node_i < NUMB_OF_NODE; node_i++)
        {
        w = ( xx[2] + xx[1] * dv[node_i] ) * invxx0;
        phi = Phi( TruncatedValue * invxx0 - w );
        if (phi > 0.0)
            {
            z = 0.0;
            for (j = 0; j < DatainGrp[ith_grp]; j++)
                {
                d = y_ij[ j ] * invxx0 - w;
                z += (d * d);
                }
            d = Node[node_i] + missing[ith_grp] * LOG( phi )
- 0.5 * z - e;

            if (d > TINY)
                s += EXP( d );
            else
                s += EP;
            }

        w = ( xx[2] - xx[1] * dv[node_i] ) * invxx0;
        phi = Phi( TruncatedValue * invxx0 - w );
        if (phi > 0.0)
            {
            z = 0.0;
            for (j = 0; j < DatainGrp[ith_grp]; j++)
                {
                d = y_ij[ j ] * invxx0 - w;
                z += (d * d);
                }
            d = Node[node_i] + missing[ith_grp] * LOG( phi )
- 0.5 * z - e;

            if (d > TINY)
                s += EXP( d );
```

```
            else
                s += EP;
            }
        }
    return 0.5*s;
    }// End of MY_TYPE ONEWAY_MM:.IntgrT(...)


    main()
    {
    ONEWAY_MM *ow = new ONEWAY_MM( obs, data_struct, -1.0 );

    ow->ShowResult( OW_MM );

    delete ow;
    return 0;
    }
```

A.2 One-Way Robust

```
    #include <assert.h>
    #include <conio.h>
    #include <math.h>
    #include <graphics.h>
    #include <stdio.h>
    #include <stdlib.h>
    #include <string.h>
    #include <time.h>


    typedef signed char   Boolean;
    typedef unsigned char UCHAR;

#ifdef EIGHTY_BITS
    #define HUGE         4900
    #define H_VAL        1.0E+4900
    #define EP           1.0E-4900
    #define TINY         -4900.0
    #define EXP(x)       expl( x )
    #define LOG(x)       logl( x )
    #define SQRT(x)      sqrtl( x )
    typedef long double MY_TYPE;
#else
    #define HUGE         300
    #define H_VAL        1.0E+300
    #define EP           1.0E-300
    #define TINY         -300.0
    #define EXP(x)       exp( x )
    #define LOG(x)       log( x )
    #define SQRT(x)      sqrt( x )
    typedef double       MY_TYPE;
#endif

    const  UCHAR      OW_ANOVA = 0x01;
    const  UCHAR      OW_MLE   = 0x02;
    const  UCHAR      OW_MM    = 0x04;
```

```
/* Parameters for robust estimate */
MY_TYPE Alpha = 0.05;
MY_TYPE Beta = 1.345;
MY_TYPE c1 = (1.0 - Alpha) / 2.5066283;
MY_TYPE C1 = (1.0 - Alpha) / (Beta * 2.5066283);
MY_TYPE C2 = C1 * EXP(0.5 * Beta * Beta);
MY_TYPE C3 = C1 * EXP(-0.5 * Beta * Beta);

#define NUMB_SEARCH_STEP     5
#define NUMB_OF_NODE       150
const float UpperBound =  20.0;
const float LowerBound = -20.0;

#define NUMB_OF_GROUP        5
#define NUMB_PER_GROUP       8

const float TruncatedValue = -0.0;
short data_struct[NUMB_OF_GROUP] = {2, 8, 1, 8, 2};

float  obs[NUMB_OF_GROUP * NUMB_PER_GROUP] =
{
1.236,  1.081,
1.624,  1.718,  2.131,  3.661,  3.352,  1.899,  2.695,  1.486,
0.110,
0.807,  1.772,  3.587,  3.119,  2.474,  3.086,  2.705,  2.573,
0.821,  1.225,
};


void fatal_err( char *msg )
{
printf( "Error: %s!\n", msg );
exit(1);
}/* end of fatal_err(...) */


MY_TYPE Erf( MY_TYPE x )
{
static MY_TYPE a[] = { 0.0705230784, 0.0422820123, 0.0092705272,
                       0.0001520143, 0.0002765672, 0.0000430638 };
MY_TYPE y = 1.0, xx = x;
short i;

for ( i=0;i<6;i++ ) {
     y += a[i]*xx;
     xx *= x;
     }
return pow( y, -16.0 );
}/* end of MY_TYPE Erf(...) for Phi(...) */


MY_TYPE Phi( MY_TYPE u )
{
if ( u>=15.0 )
    return 1.0;
if ( u<=-15.0 )
    return 0.0;

if ( u>0.0 )
```

```
    return 0.5*( 2.0 - Erf( u*0.7071067812 ) );
return 0.5*Erf( -u*0.7071067812 );
}/* end of MY_TYPE Phi(...) */


class ONEWAY
{
public:
    ONEWAY( float *_Obs, short *_DataStruct );
    ~ONEWAY( void );
    virtual void SetData( float *_Obs, short *_DataStruct );
    float   *GetResult( UCHAR opt=OW_ANOVA );
    void    ShowResult( UCHAR opt=OW_ANOVA|OW_MLE );
protected:
    virtual void  DOIT( void );
    void    anova( void );
    float   mle( float *xx );
    float   Optimize( float *lx, float *dx );
    char    *FileName, IsDONE;
    short   NoOfData, *DatainGrp;
    float   *Data;
    float   x[ 10 ];
    FILE    *in;
};


ONEWAY::ONEWAY( float *_Obs, short *_ObsinGrp )
{
DatainGrp = NULL;
Data = NULL;
SetData( _Obs, _ObsinGrp );
FileName = NULL;
IsDONE = 0;
}//End of ONEWAY:.ONEWAY(...)


ONEWAY::~ONEWAY( void )
{
delete [] Data;
delete [] DatainGrp;
if ( FileName )
    delete [] FileName;
else
    fclose( in );
}//End of ONEWAY::~ONEWAY()


void ONEWAY::SetData( float *_Obs, short *_ObsinGrp )
{
if ( DatainGrp )
    delete [] DatainGrp;
DatainGrp = new short[ NUMB_OF_GROUP ];
if (!DatainGrp)
    fatal_err( "No Memory" );

NoOfData = 0,
for ( short i=0;i<NUMB_OF_GROUP;i++ ) {
    DatainGrp[i] = _ObsinGrp[ i ];
    NoOfData += DatainGrp[ i ];
```

```
    }
if ( Data )
    delete [] Data;

Data = new float[ NoOfData ];
if (!Data)
    fatal_err( "No Memory" );

for ( i=0;i<NoOfData;i++ )
    Data[ i ] = _Obs[ i ];
IsDONE = 0;
}//End of ONEWAY::SetData(...)


float *ONEWAY::GetResult( UCHAR opt )
{
if ( !IsDONE )
     DOIT();
if ( opt&OW_ANOVA )
    return x;
if ( opt&OW_MLE )
    return &x[3];
else
    return &x[6];
}//End of ONEWAY::GetResult(...)


void ONEWAY::ShowResult( UCHAR opt )
{
if ( !IsDONE )
    DOIT();
printf("ANOVA sigma_o %f  sigma_1 %f  mu %f\n", x[0], x[1], x[2] );
printf("MLE   sigma_o %f  sigma_1 %f  mu %f\n", x[3], x[4], x[5] );
if ( opt&OW_MM )
    printf("MM    sigma_o %f  sigma_1 %f  mu %f\n", x[6], x[7],
   x[8] );
}//End of ONEWAY::ShowResult(...)


void ONEWAY::DOIT( void )
{
short i;
float lx[4], ux[4], dx[4], oldoptf, optf, c;

IsDONE = 1;
anova();
lx[0] = lx[1] = lx[2] = 0.01;
for ( i=0;i<3;i++ ) {
    ux[i] = x[ i ]*2.0;
    dx[i] = ( ux[i] - lx[i] )/NUMB_SEARCH_STEP;
    }

oldoptf = 1.0; optf = 0.0;
while ( fabs( optf - oldoptf ) > 0 0001 ) {
    oldoptf = optf;
    optf = Optimize( lx, dx );
    for ( i=0;i<3;i++ ) {
        c = (ux[i] - lx[i])/NUMB_SEARCH_STEP;
        lx[i] = x[ 3+i ] - c;
        if ( lx[i]<0.01 )
```

```
                lx[i] = 0.01;
            ux[i] = x[ 3+i ] + c;
            dx[i] = (ux[i] - lx[i])/NUMB_SEARCH_STEP;
            }
        }
    }//End of ONEWAY::DOIT()


    void ONEWAY::anova( void )
    {
    short i, j, k;
    float s1, s2, s3, a;
    k = 0;
    s1 = s2 = s3 = x[0] = x[1] = 0.0;
    for ( i=0;i<NUMB_OF_GROUP;i++ ) {
        x[0] += DatainGrp[i];
        x[1] += DatainGrp[i]*DatainGrp[i];
        a = 0.0;
        for ( j=0;j<DatainGrp[i];j++ ) {
            s1 += obs[ k+j ]*obs[ k+j ];
            a += obs[ k+j ];
            }
        k += DatainGrp[i];
        s2 += a*a/DatainGrp[i];
        s3 += a;
        }
    x[5] = x[2] = s3/x[0];
    s3 *= s3;
    s3 /= x[0];
    a =( x[0]*(NUMB_OF_GROUP-1) )/( x[0]*x[0] - x[1] );
    x[0] = ( s1 - s2 )/( x[0] - NUMB_OF_GROUP );
    x[1] = a*( ( s2 - s3 )/( NUMB_OF_GROUP - 1 ) - x[0] );
    }//End of ONEWAY::anova()


    float ONEWAY::mle( float *xx )
    {
    short  j, i, l;
    float  a, b, c, d, e, f;
    a = 1.0/xx[0];
    f = l = 0;
    for ( i=0;i<NUMB_OF_GROUP;i++ ) {
        f += log( pow( xx[0], DatainGrp[i]-1 )*
  ( xx[0] + DatainGrp[i]*xx[1] ) );
        b = xx[1]/( xx[0]*xx[0] + DatainGrp[i]*xx[0]*xx[1] );
        d = e = 0.0;
        for ( j=0;j<DatainGrp[i];j++ ) {
            c = Data[ l+j ] - xx[ 2 ];
            d += c*c;
            e += c;
            }
        f += a*d - b*e*e;
        l += DatainGrp[i];
        }
    return f;
    }//End of ONEWAY::mle()


    float  ONEWAY::Optimize( float *lx, float *dx )
    {
```

```
float  xx[ 4 ], f, optf;
short  j, i, k;

optf = 1.0E+8;
xx[0] = lx[0];
for ( i=0;i<=NUMB_SEARCH_STEP;i++ ) {
  /* loop for sigma_o */
    xx[1] = lx[1];
    for ( j=0;j<=NUMB_SEARCH_STEP;j++ ) {
  /* loop for sigma_1 */
        xx[2] = lx[2];
        for ( k=0; k<=NUMB_SEARCH_STEP;k++) {
            f = mle( xx );
            if ( f<optf ) {
                optf = f;
                x[ 3 ] = xx[ 0 ];
                x[ 4 ] = xx[ 1 ];
                x[ 5 ] = xx[ 2 ];
                }
            xx[2] += dx[2];
            }
        xx[1] += dx[1];
        }
    xx[0] += dx[0];
    }
return optf;
}//End of ONEWAY::Optimize(...)


class ONEWAY_MM : public ONEWAY
{
public:
    ONEWAY_MM( float *_Obs, short *_ObsinGrp, float _TruncateV );
    ~ONEWAY_MM( void );
    void   DOIT( void );
protected:
    float  Optimize( float *lx, float *dx );
    void   SetNodes( void );
    float  func( float *xx );
    MY_TYPE IntgrT( float *xx, short ith_grp );
    short  *missing, *GrpPtr;
    float  TruncatedValue;
    MY_TYPE interval;
};


ONEWAY_MM::ONEWAY_MM( float *_Obs, short *obs_per_grp,
float _TruncatedV ) :
    ONEWAY( _Obs, obs_per_grp )
{
missing = new short[ NUMB_OF_GROUP ];
GrpPtr = new short[ NUMB_OF_GROUP ];

if (!missing || !GrpPtr)
    fatal_err( "No Memory" );

for ( short i=0;i<NUMB_OF_GROUP;i++ )
    missing[i] = NUMB_PER_GROUP - obs_per_grp[i];
```

```
GrpPtr[0] = 0;
for ( i=1;i<NUMB_OF_GROUP;i++ )
    GrpPtr[i] = GrpPtr[i - 1] +  obs_per_grp[i - 1];

TruncatedValue = _TruncatedV;
SetNodes();
}//End of ONEWAY_MM::ONEWAY_MM(...)


ONEWAY_MM::~ONEWAY_MM( void )
{
delete [] GrpPtr;
delete [] missing;
}//End of ONEWAY_MM::~ONEWAY_MM()


void ONEWAY_MM::DOIT( void )
{
short i;
float lx[4], ux[4], dx[4], oldoptf, optf, c;

ONEWAY::DOIT();

lx[0] = lx[1] =  lx[2] = 0.01;
for ( i=0;i<3;i++ ) {
    ux[i] = x[ 3+i ]*2.0;
    dx[i] = ( ux[i] - lx[i] )/NUMB_SEARCH_STEP;
    }

oldoptf = 1.0; optf = 0.0;
while ( fabs( optf - oldoptf ) > 0.0005 ) {
    oldoptf = optf;
    optf = Optimize( lx, dx );

    for ( i=0;i<3;i++ )
        {
        printf( " %.6f", x[6 + i] );
        c = (ux[i] - lx[i])/NUMB_SEARCH_STEP;
        lx[i] = x[ 6+i ] - c;
        if ( lx[i]<0.01 )
            lx[i] = 0.01;
        ux[i] = x[ 6+i ] + c;
        dx[i] = (ux[i] - lx[i])/NUMB_SEARCH_STEP;
        }
    printf( "\n" );
    }
}//End of ONEWAY_MM::DOIT()


float  ONEWAY_MM::Optimize( float *lx, float *dx )
{
float  xx[ 4 ], f, optf;
short  j, i, k;

optf = 1.0E+8;
xx[2] = lx[2];
for ( k=0;k<=NUMB_SEARCH_STEP;k++ ) {
    xx[0] = lx[0];
    for ( i=0;i<=NUMB_SEARCH_STEP;i++ ) {
/* loop for sigma_o */
```

```
        xx[1] = lx[1];
        for ( j=0;j<=NUMB_SEARCH_STEP;j++ ) {
/* loop for sigma_1 */
            f = func( xx );
            if ( f<optf ) {
                optf = f;
                x[ 6 ] = xx[ 0 ];
                x[ 7 ] = xx[ 1 ];
                x[ 8 ] = xx[ 2 ];
                }
            xx[1] += dx[1];
            }
        xx[0] += dx[0];
        }
    xx[2] += dx[2];
    }
return optf;
}//End of ONEWAY_MM::Optimize(...)


void ONEWAY_MM::SetNodes( void )
{
interval = (UpperBound - LowerBound) / NUMB_OF_NODE;
}// End of void ONEWAY_MM::SetNodes()


float ONEWAY_MM::func( float *xx )
{
short i;
float f = 0.0;
MY_TYPE ff;

xx[0] = SQRT( xx[0] );
xx[1] = SQRT( xx[1] );

for ( i=0; i<NUMB_OF_GROUP;i++ )
     {
     ff = IntgrT( xx, i );
     if (ff > EP)
         f -= LOG( ff );
     else
         f -= TINY;
     }

xx[0] *= xx[0];
xx[1] *= xx[1];
return f;
}//End of ONEWAY_MM::func(...)


MY_TYPE ONEWAY_MM::IntgrT( float *xx, short ith_grp )
{
short  j, node_i;
float * y_ij = &Data[ GrpPtr[ith_grp] ];
MY_TYPE d, f1, f2, f3;      /* variables */
MY_TYPE s;                        /* result */
MY_TYPE w;
MY_TYPE x;                  /* x for f(x) */
MY_TYPE invxx0, invxx1;
```

```
invxx0 = 1.0 / xx[0];
invxx1 = 1.0 / xx[1];

s = 0.0;
x = LowerBound;
for (node_i = 0; node_i <= NUMB_OF_NODE; node_i++)
    {
    w = Beta * (TruncatedValue - x) * invxx0;

    f1 = 1.0;
    for (j = 0; j < DatainGrp[ith_grp]; j++)
        {
        d = (y_ij[j] - x) * invxx0;
        if (x - xx[0] * Beta <= y_ij[j] && y_ij[j] <= x
 + xx[0] * Beta)
            f1 *= c1 * invxx0 * EXP( -0.5 * d * d );
        else
            f1 *= c1 * invxx0 * EXP( 0.5 * Beta * Beta
   - Beta * fabs( d ));
        }

    if (0 != missing[ith_grp])
        {
        if (TruncatedValue <= x - Beta * xx[0])
            d = C1 * EXP( 0.5 * Beta * Beta + w );
        else if ( x - Beta * xx[0] < TruncatedValue
              && TruncatedValue <= x + Beta * xx[0])
            d = C3 + (1.0 - Alpha) * (Phi( w / Beta) +
Phi( Beta ) - 1.0);
        else
            {
            // if ( TruncatedValue  => x + Beta * xx[0] )
            d = 2.0 * C3 + (1.0 - Alpha) * (2.0 * Phi( Beta )
 - 1.0) - C2 * EXP(- w);
            }

        f2 = pow( d, (MY_TYPE) missing[ith_grp] );
        }
    else
        f2 = 1.0;

    d = (x - xx[2]) * invxx1;
    f3 = EXP(-0.5 * d * d) * 0.7071067 * invxx1;
    s += interval * f1 * f2 * f3;

    x += interval;
    }

return s;
}// End of MY_TYPE ONEWAY_MM::IntgrT(...)


main()
{
ONEWAY_MM *ow = new ONEWAY_MM( obs, data_struct, TruncatedValue );

ow->ShowResult( OW_MM );

delete ow;
```

```
    return 0;
    }
```

A.3 Two-Way Nested

There are three source file involved in this programme.

A.3.1 Header File

```
    #include <stdio.h>


#ifdef EIGHTY_BITS
    #define HUGE        4900
    #define H_VAL       1.0E+4900
    #define EP          1.0E-4900
    #define TINY        -4900.0
    #define EXP(x)      expl( x )
    #define LOG(x)      logl( x )
    typedef long double MY_TYPE;
#else
    #define HUGE        300
    #define H_VAL       1.0E+300
    #define EP          1.0E-300
    #define TINY        -300.0
    #define EXP(x)      exp( x )
    #define LOG(x)      log( x )
    typedef double      MY_TYPE;
#endif

    #define MAXDIM        100

    #define NUMB_OF_SIR    8
    #define NUMB_OF_DAM    3
    #define NUMB_OF_SIB   20

    typedef  unsigned char   UCHAR;
    typedef  unsigned short  USHORT;
    typedef  char            Boolean;

    const USHORT    MLEMode           = 0x0001;
    const USHORT    MissSirIncl       = 0x0002;
    const USHORT    MissDamIncl       = 0x0004;


    const USHORT    BadMemory         = 0x0001;
    const USHORT    BadFile           = 0x0002;

    Boolean PlotResults(float *obs,int nobs, float lb, float ub);


    class UTW_ANOVA             // unbalanced two-way
analysis of variance
    {
    public:
```

```
        UTW_ANOVA( char *DataFileName );
        ~UTW_ANOVA( void );
        float *GetResults( void );
    protected:
        virtual void DoIt( void );
        short  NoOfSir, NoOfDam, NoOfOffspring;
        short  *SirToDam, *DamToOffspring;
        float  *Obs, x[10];
        Boolean IsDone;
        FILE   *in;
    };

    class UTW_MLE : public UTW_ANOVA //unbalanced two-way
      max. likelihood est.
    {
    public:
        UTW_MLE( char *DataFile, USHORT _flag=0U ):
            UTW_ANOVA( DataFile ), SearchStep(4), flag(_flag){}
        void   DoANOVA( void );
        void   GetPlotData(float *g,short n,float *x,float
   lx,float ux, short Obs);
        USHORT GetFlag( void ) { return flag; }
        void   SetFlag( USHORT _flag ) { flag=_flag; }
    protected:
        virtual MY_TYPE func( float *xx );
        USHORT  flag;
    private:
        short   SearchStep;
    };

    class UTW_MM : public UTW_MLE
    {
    public:
        UTW_MM(char *DataFile, USHORT _flag=0U);
        ~UTW_MM( void );
        const float * GetTargetParameters(void)
            { return &target_parameters[0]; }
    protected:
        MY_TYPE func( float *xx );
    private:
        MY_TYPE IntgrV( float *xx, short ithSir, short Dam_i );
        MY_TYPE IntgrT( MY_TYPE dv, float *xx, short miss,
short Dam_ij );
        MY_TYPE IntgrV( float *xx );
        MY_TYPE IntgrT( MY_TYPE dv, float *xx );
        void    SetNodes( void );
        MY_TYPE Node[10], dt[10];
        float  TruncatedValue, *Sum_Yij, target_parameters[5];
        short  *missing, *Obsptr, NoOfNode;
        short  FullNoOfSir, FullNoOfDam, FullNoOfSib;
    };

    class TWG
    {
    public:
```

```
    TWG( float _sigma0, float _sigma1, float _sigma2, float _mean,
         float _TrimRate=0.25, short _RandSeed=0 );
    ~TWG( void );
    Boolean DoIt( void );
    void  SetTrimRate( float _TrimRate );
    void  Write( char *FileName=NULL );
    void  WriteMis( char *FileName=NULL );
private:
    void  GenerateIt( void );
    short Trim( void );
    float NrmlGen( float _var );
    void  sort_f( float *x, short n );
    float *Sir, *Dam[MAXDIM];
    float *Obs[MAXDIM][MAXDIM], *Tmp;
    float sigma0, sigma1, sigma2, mean, tv, TrimRate;
    UCHAR *SirToDam, *SirDamChild[MAXDIM];
    UCHAR *sirtodam, *sirdamchild[MAXDIM], noofsir;
    short NoOfSir, NoOfDam, NoOfOffspring, NoTrimmed;
    short RandSeed;
    FILE  *io;
};

void   Fatal_Error( char *msg );
extern USHORT TW_Error;
```

A.3.2 Two Source Files

This is the primary source code for computation.

```
#include "mutw.hpp"

#include <conio.h>
#include <graphics.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define TRUE       1
#define FALSE      0


USHORT TW_Error;


void Fatal_Error( char *msg )
{
printf( "Error: %s!\n", msg );
putch(7);
exit(1);
}/* end of Fatal_Error(...) */


MY_TYPE Erf( MY_TYPE x )
{
static MY_TYPE a[] = { 0.0705230784, 0.0422820123, 0.0092705272,
                       0.0001520143, 0.0002765672, 0.0000430638 };
```

```
MY_TYPE y = 1.0, xx = x;
short i;

for ( i=0;i<6;i++ ) {
     y += a[i]*xx;
     xx *= x;
     }
return pow( y, -16.0 );
}/* end of MY_TYPE Erf(...) for Phi(...) */


MY_TYPE Phi( MY_TYPE u )
{
if ( u>=15.0 )
    return 1.0;
if ( u<=-15.0 )
    return 0.0;

if ( u>0.0 )
    return 0.5*( 2.0 - Erf( u*0.7071067812 ) );
return 0.5*Erf( -u*0.7071067812 );
}/* end of MY_TYPE Phi(...) */


MY_TYPE phi( MY_TYPE w )
{
return ( 0.398942280401433 * EXP( -0.5 * w * w ) );
}                                   // phi()


MY_TYPE phi_deriv( MY_TYPE w )
{
return ( -w * phi( w ) );
}                                   // phi_deriv()


TWG::TWG( float _sigma0, float _sigma1, float _sigma2,
          float _mean, float _TrimRate, short _RandSeed ):
          sigma0(_sigma0), sigma1(_sigma1), sigma2(_sigma2),
          mean(_mean), RandSeed(_RandSeed)
{
short i, j, k;

if ( RandSeed<=0 ) {
    randomize();
    RandSeed = rand();
    }
srand( RandSeed );
SetTrimRate( _TrimRate );
//   NoOfSir = 5; // + random( 6 );
NoOfSir = NUMB_OF_SIR;
SirToDam = new UCHAR[ NoOfSir ];
sirtodam = new UCHAR[ NoOfSir ];
Sir = new float[ NoOfSir ];

NoOfDam = NoOfOffspring = 0;
for ( i=0;i<NoOfSir;i++ ) {
//       j = 5; // + random( 4 );
    j = NUMB_OF_DAM;
    SirDamChild[i] = new UCHAR[ j ];
```

```
    sirdamchild[i] = new UCHAR[ j ];
    Dam[i] = new float[ j ];
    NoOfDam += j;
    SirToDam[i] = j;
    }

for ( i=0;i<NoOfSir;i++ )
    for ( j=0;j<SirToDam[i];j++ ) {
//          k = 16; // + random( 15 );
        k = NUMB_OF_SIB;
        NoOfOffspring += k;
        Obs[i][j] = new float[ k ];
        SirDamChild[i][j] = k;
        }
Tmp = new float[ NoOfOffspring ];
}// End of TWG::TWG(...)


TWG::~TWG( void )
{
short i, j;
delete [] Tmp;
for ( i=NoOfSir-1;i>=0;i-- )
    for ( j=SirToDam[i]-1;j>=0;j-- )
         delete [] Obs[i][j];

for ( i=NoOfSir-1;i>=0;i-- ) {
    delete [] Dam[i];
    delete [] sirdamchild[i];;
    delete [] SirDamChild[i];
    }
delete [] Sir;
delete [] sirtodam;
delete [] SirToDam;
}// End of TWG::~TWG()


void TWG::SetTrimRate( float _TrimRate )
{
TrimRate = _TrimRate;
if ( TrimRate<0.0 )
    TrimRate = 0.0;
if ( TrimRate>0.95 )
    TrimRate = 0.95;
}//End of TWG::SetTrimRate(...)


Boolean TWG::DoIt( void )
{
short j = 0;
float c;

do  {
    GenerateIt();
    c = NoOfOffspring;
    c *= TrimRate;
    NoTrimmed = (short)c;
    if ( NoTrimmed<1 )
        tv = Tmp[0] - 0.1;
```

```
    else
        tv = Tmp[ NoTrimmed-1 ];
    j ++;
    }  while ( Trim()==FALSE && j<100 );

if ( j>=100 )
    return FALSE;
Write();
WriteMis();
return TRUE;
}// End of TWG::DoIt(...)


void TWG::GenerateIt( void )
{
short kkk, k, j, i;

for ( i=0;i<NoOfSir;i++ )
    Sir[i] = NrmlGen( sigma1 );

for ( i=0;i<NoOfSir;i++ )
    for ( j=0;j<SirToDam[i];j++ )
        Dam[i][j] = NrmlGen( sigma2 );

kkk = 0;
for ( i=0;i<NoOfSir;i++ )
    for ( j=0;j<SirToDam[i];j++ )
        for ( k=0;k<SirDamChild[i][j];k++ )
            Tmp[kkk++] = Obs[i][j][k] =
                mean + Sir[i] + Dam[i][j] + NrmlGen( sigma0 );

sort_f( Tmp, NoOfOffspring );
}// End of TWG::GenerateIt()


short TWG::Trim()
{
short i, j, k;

noofsir = NoOfSir;
for ( i=0;i<NoOfSir;i++ ) {
    sirtodam[i] = SirToDam[i];
    for ( j=0;j<SirToDam[i];j++ )
        sirdamchild[i][j] = SirDamChild[i][j];
    }

for ( i=0;i<NoOfSir;i++ )
    for ( j=0;j<SirToDam[i];j++ )
        {
        for ( k=0;k<SirDamChild[i][j];k++ )
            if ( Obs[i][j][k]<=tv )
                sirdamchild[i][j] --;
        if ( sirdamchild[i][j]<2 )
            {
            sirdamchild[i][j] = 0;
            sirtodam[i] --;
            }
        }
```

```
for ( i=0;i<NoOfSir;i++ )
    if ( sirtodam[i]<2 ) {
        sirtodam[i] = 0;
        noofsir --;
        }

if ( noofsir<2 )
    return FALSE;
return TRUE;
}// End of TWG::Trim(...)


float TWG::NrmlGen( float var )
{
float b = 1.0 + rand();
float c = 1.0 + rand();
b /= 32767.0;
c /= 32767.0;
return( sqrt(-2.0*log(b))*cos(6.2831853*c)*sqrt(var) );
}/* end of normal random number generation */


void TWG::Write( char *FileName )
{
short i, j, k;
float min, max;

if ( FileName )
    io = fopen( FileName, "w+t" );
else
    io = fopen( "tw.dat", "w+t" );
fprintf( io, " %2d\n", NoOfSir );

for ( i=0;i<NoOfSir;i++ )
    fprintf( io, " %2d", SirToDam[i] );
fprintf( io, "\n" );

for ( i=0;i<NoOfSir;i++ ) {
    for ( j=0;j<SirToDam[i];j++ )
        fprintf( io, " %2d", SirDamChild[i][j] );
    fprintf( io, "\n" );
    }

min = 1.0E+30;
max = -1.0E+30;
for ( i=0;i<NoOfSir;i++ )
    for ( j=0;j<SirToDam[i];j++ ) {
        for ( k=0;k<SirDamChild[i][j];k++ )
            {
            fprintf( io, " %6.2f", Obs[i][j][k] );
            if ( min > Obs[i][j][k] )
                min = Obs[i][j][k];
            if ( max < Obs[i][j][k] )
                max = Obs[i][j][k];
            }
        fprintf( io, "\n" );
        }
for ( i=0;i<NoOfSir;i++ )
    for ( j=0;j<SirToDam[i];j++ )
```

```
        fprintf( io, "0\n" );

fprintf( io, "%d %d %d\n", NUMB_OF_SIR, NUMB_OF_DAM, NUMB_OF_SIB );
fprintf( io, "%.2f %.2f %.2f %.2f\n", sigma0, sigma1, sigma2,
 mean );
fprintf( io, "%.2f\n", min - 0.1 );
fprintf( io, "-----------------------------\n" );
fprintf( io, "FullSir: %d  FullDam: %d  FullSib: %d\n",
NUMB_OF_SIR, NUMB_OF_DAM, NUMB_OF_SIB );
fprintf( io, "min: %.2f  max: %.2f  TotalData: %d\n", min, max,
NoOfOffspring );
fprintf( io, "RandSeed: %d\n", RandSeed );
fprintf( io, "sigma0^2: %.2f  sigma1^2: %.2f  sigma2^2: %.2f
mean:%.2f\n", sigma0, sigma1, sigma2, mean );
fclose( io );
}// End of TWG::Write(...)


void TWG::WriteMis( char *FileName )
{
short i, j, k;

if ( FileName )
    io = fopen( FileName, "w+t" );
else
    io = fopen( "tw.mis", "w+t" );
fprintf( io, " %2d\n", noofsir );

for ( i=0;i<NoOfSir;i++ )
    if ( sirtodam[i]>0 )
        fprintf( io, " %2d", sirtodam[i] );
fprintf( io, "\n" );

for ( i=0;i<NoOfSir;i++ )
    {
    if ( sirtodam[i]>0 )
        {
        for ( j=0;j<SirToDam[i];j++ )
            if ( sirdamchild[i][j]>0 )
                fprintf( io, " %2d", sirdamchild[i][j] );
        if ( sirtodam[i]>0 )
            fprintf( io, "\n" );
        }
    }

for ( i=0;i<NoOfSir;i++ )
    if ( sirtodam[i]>0 )
        {
        for ( j=0;j<SirToDam[i];j++ )
            {
            if ( sirdamchild[i][j]>0 )
                {
                for ( k=0;k<SirDamChild[i][j];k++ )
                    if ( Obs[i][j][k]>tv )
                        fprintf( io, " %6.2f", Obs[i][j][k] );
                fprintf( io, "\n" );
                }
```

```
            }
        }
   for ( i=0;i<NoOfSir;i++ )
       for ( j=0;j<SirToDam[i];j++ )
           if ( sirtodam[i]>0 && sirdamchild[i][j]>0 )
               fprintf( io, "%2d\n", SirDamChild[i][j]
-sirdamchild[i][j] );

   fprintf( io, "%d %d %d\n", NUMB_OF_SIR, NUMB_OF_DAM, NUMB_OF_SIB );
   fprintf( io, "%.2f %.2f %.2f %.2f\n", sigma0, sigma1, sigma2,
    mean );
   fprintf( io, "%.2f\n", tv );
   fprintf( io, "------------------------------\n" );
   fprintf( io, "Before Trimmed -- Sir: %d  Dam: %d  Sib: %d\n",
   NUMB_OF_SIR, NUMB_OF_DAM, NUMB_OF_SIB );
   fprintf( io, "TruncatedValue %.2f  TrimRatio: %.2f  TotalData:
   %d\n", tv, TrimRate, NoOfOffspring-NoTrimmed );
   fprintf( io, "sigma0^2: %.2f  sigma1^2: %.2f  sigma2^2: %.2f
   mean:%.2f\n", sigma0, sigma1, sigma2, mean );
   fclose( io );
   }// End TWG::WriteMis(...)

   void TWG::sort_f(float *ra, short n)
   {
   unsigned   l,j,ir,i;
   float      rra;

   l=(n >> 1)+1;
   ir=n;
   for (;;) {
       if (l > 1)
           rra=*(ra+(--l)-1);
       else {
           rra=*(ra+ir-1);
           *(ra+ir-1)=*(ra);
           if (--ir == 1) {
               *(ra)=rra;
               return;
               }
           }
       i=l;
       j=l << 1;
       while (j <= ir) {
           if (j < ir && *(ra+j-1) < *(ra+j)) ++j;
           if (rra < *(ra+j-1)) {
               *(ra+i-1)=*(ra+j-1);
               j += (i=j);
               }
           else j=ir+1;
           }
      *(ra+i-1)=rra;
      }
   }/* end of TWG::sort_float(...) */

   Boolean PlotResults(float *obs,int nobs, float lb, float ub)
```

```
{
int   x1,y1,x2,y2,i,j,k,l,ll, nyaxi=20;
float ystp, ymin, ymax, a, hstp, vstp, v, vv;
char  p[20];
int   graphdriver=DETECT, graphmode;

registerbgidriver(EGAVGA_driver);
initgraph(&graphdriver,&graphmode,"");
cleardevice();
x1=0.1*getmaxx();y1=0.1*getmaxy(); x2=9*x1; y2=9*y1;
hstp=x1*8.0; hstp /= (nobs-1);vstp=y1*8.0/nyaxi;

ymax=-1.5E8;ymin=1.5E8;
for (i=0;i<nobs;i++) {
    a=*(obs+i);
    if (a>ymax) ymax=a;
    if (a<ymin) ymin=a;
    }

if ( fabs( ymax-ymin )<1.0E-16 ) {
    closegraph();
    return FALSE;
    }

ystp=(ymax-ymin)/nyaxi;
a=8*y1-2.;
a/=(ymax-ymin);

if ( obs[0] != ymin && ymin != obs[i - 1] )
    {
    line(x1, y2 - 1.92 * a - 1, x2, y2 - 1.92 * a - 1);

    ymax = ub-lb;
    ymax /= nobs;

    i = 0;
    while (i < nobs)
        {
        if ( obs[i] <= ymin + 1.92 )
            {
            v = x1 + + ((float)i - 1.0) * hstp + hstp *
  (ymin + 1.92 - obs[i - 1])
                / ( obs[i] - obs[i - 1] );

            line( v, y1, v, y2 );

            vv  = lb + ((float)i - 1.0) * ymax + ymax *
  (ymin + 1.92 - obs[i - 1])
                / ( obs[i] - obs[i - 1] );
            sprintf( p,"%.2f", vv );
            outtextxy( v + 10, y2 - 2.0 * a - 20, p );

            break;
            }
       i ++;
       }

    while (i < nobs)
```

```
            if ( obs[i ++] == ymin )
                break;

        while (i < nobs)
            {
            if ( obs[i] >= ymin + 1.92 )
                {
                v = x1 + ((float)i - 1.0) * hstp + hstp *
(ymin + 1.92 - obs[i - 1])
                    / ( obs[i] - obs[i - 1] );
                line( v, y1, v, y2 );

                vv  = lb + ((float)i - 1.0) * ymax + ymax *
(ymin + 1.92 - obs[i - 1])
                    / ( obs[i] - obs[i - 1] );
                sprintf( p,"%.2f", vv );
                outtextxy( v + 10, y2 - 2.0 * a - 20, p );

                break;
                }
            i ++;
            }
       }

    for (i=0;i<nobs;i++)
        *(obs+i)=(*(obs+i)-ymin)*a;

    setlinestyle(0,0xffff,1);
    setcolor(15);
    rectangle(x1,y1,x2,y2);
     /* draw frame */

    for (i=0;i<=nyaxi;i++) {
     /* draw vertical axis & scales */
        l=4;j=y1+vstp*i;
        if (i%5==0) {
            l=6;if(i%10==0) l=8;
            sprintf(p,"%.2f",ymin+(nyaxi-i)*ystp);
            outtextxy(1,j,p);
            }
        line(x1-l,j,x1,j);
        }

    k=1+nobs/12;
    v = ub-lb;
    v /=nobs;
    for (i=0; i<nobs;i+=k) {
      /* draw horizontal axis & scales */
        l=5;j=x1+hstp*i;
        sprintf(p,"%.2f", lb+i*v);
        outtextxy(j-15,y2+0.125*x1,p);
        line(j,y2,j,y2+l);
        }

    line(x2,y2,x2,y2+l);
    sprintf(p,"%.2f",ub);
    outtextxy(x2-5,y2+0.125*x1,p);
```

```
sprintf( p, "Min at = %.3f", obs[nobs] );
outtextxy( 4*x1, y1 - 20, p );

setcolor(10);
a = k = x1;
for ( j=1; j<nobs; j++ )
    {
    l = y2-*( obs+j-1 );
    ll = y2-*( obs+j );
    line( k, l, k+hstp, ll );
    a += hstp;
    k = a;
    }

getch();
closegraph();
return TRUE;
}/* end of function linecht */


UTW_ANOVA::UTW_ANOVA( char *DataFile )
{
short i, j;
float c;

in = fopen( DataFile, "rt" );
fscanf( in, "%d", &NoOfSir );
if ( NoOfSir<2 )
    Fatal_Error("(UTW_ANOVA) Illegal NoOfSir" );
SirToDam = new short[ NoOfSir ];
if ( SirToDam==NULL )
    Fatal_Error("UTW_ANOVA): Memory Allocation(SirToDam)" );
NoOfDam = i = 0;
while ( i<NoOfSir) {
    if ( feof( in ) )
        Fatal_Error("(UTW_ANOVA): DataFile incorrect(SirToDam)" );
    fscanf( in, "%d", &j );
    if ( j<1 || j>1000 )
        Fatal_Error("(UTW_ANOVA): Illegal NoOfDam" );
    SirToDam[i] = j;
    NoOfDam += j;
    i ++;
    }
DamToOffspring = new short[ NoOfDam ];
if ( DamToOffspring==NULL )
    Fatal_Error("(UTW_ANOVA): Memory Allocation(DamToOffspring)" );
NoOfOffspring = i = 0;
while ( i<NoOfDam ) {
    if ( feof( in ) )
        Fatal_Error("(UTW_ANOVA):
 DataFile incorrect(DamToOffspring)" );
    fscanf( in, "%d", &j );
    if ( j<1 || j>1000 )
        Fatal_Error("(UTW_ANOVA): Illegal NoOfOffspring" );
    DamToOffspring[i] = j;
    NoOfOffspring += j;
    i ++;
    }
```

```
Obs = new float[ NoOfOffspring ];
if ( Obs==NULL )
    Fatal_Error("(UTW_ANOVA): Memory Allocation(Obs)" );
i = 0;
while ( i<NoOfOffspring ) {
    if ( feof( in ) )
        Fatal_Error("(UTW_ANOVA): DataFile incorrect(Offspring)" );
    fscanf( in, "%f", &c );
    Obs[ i ] = c;
    i ++;
    }
IsDone = FALSE;
}// End of UTW_ANOVA::UTW_ANOVA(...)


UTW_ANOVA::~UTW_ANOVA( void )
{
fclose( in );
delete [] Obs;
delete [] DamToOffspring;
delete [] SirToDam;
}// End of UTW_ANOVA::~UTW_ANOVA()


void UTW_ANOVA::DoIt( void )
{
short  i, j, k, m, mm, n;
MY_TYPE GrandTotal, SS, SSsubgr, SSgroups, CT, a, b, c;
MY_TYPE MSgroups, MSsubgr, MSwithin;
MY_TYPE q2, q3, q4;

GrandTotal = SS = SSsubgr = SSgroups = CT = q2 = q3 = q4 = 0.0;
m = mm = 0;
for ( i=0;i<NoOfSir;i++ ) {
    a = 0.0;
    n = 0;
    for ( j=0;j<SirToDam[i];j++ ) {
        b = 0.0;
        for ( k=0;k<DamToOffspring[mm];k++ ) {
            b += Obs[m];
            SS += Obs[m]*Obs[m];
            m ++;
            }
        SSsubgr += b*b/k;
        a += b;
        n += k;
        mm ++;
        }
    SSgroups += a*a/n;
    GrandTotal += a;
    }
CT = GrandTotal*GrandTotal/NoOfOffspring;
m = 0;
for ( i=0;i<NoOfSir;i++ ) {
    a = b = 0.0;
    for ( j=0;j<SirToDam[i];j++ ) {
        a += DamToOffspring[m];
        b += DamToOffspring[m]*DamToOffspring[m];
        m ++;
        }
```

```
    q2 += b;
    q3 += a*a;
    q4 += b/a;
    }
MSgroups = (SSgroups-CT)/(NoOfSir-1);
MSsubgr = (SSsubgr-SSgroups)/(NoOfDam-NoOfSir);
MSwithin = (SS-SSsubgr)/(NoOfOffspring-NoOfDam);
a = ( q4 - q2/NoOfOffspring )/(NoOfSir-1);
b = -q4;
b += NoOfOffspring;
b /= NoOfDam-NoOfSir;
c = -q3;
c /= NoOfOffspring;
c += NoOfOffspring;
c /= NoOfSir-1;
x[0] = MSwithin;
x[2] = (MSsubgr-MSwithin)/b;
x[1] = (MSgroups-MSwithin-a*x[2])/c;
x[3] = GrandTotal/NoOfOffspring;
IsDone = TRUE;
}// End of void UTW_ANOVA::DoIt()


float *UTW_ANOVA::GetResults( void )
{
if ( !IsDone )
    DoIt();
return &x[0];
}// End of float *UTW_ANOVA::GetResults()


MY_TYPE UTW_MLE::func( float *xx )
{
short  k, io, j, id, i, m, ic;
MY_TYPE b, c, aa, bb, bbb, d, u, vo, v1, v2, z, det;

u = xx[3];
id = io = 0;
det = vo = v1 = v2 = 0.0;
for ( i=0;i<NoOfSir;i++ ) {
    m = -SirToDam[i];
    aa = 1.0;
    bb = bbb = 0.0;
    for ( j=0;j<SirToDam[i];j++ ) {
        ic = DamToOffspring[ id++ ];
        m += ic;
        c = xx[0] + ic*xx[2];
        z = 1.0/c;
        aa *= c;
        bb += ic*z;
        b = 0.0;
        for ( k=0;k<ic;k++ ) {
            d = Obs[io++] - u;
            vo += d*d;
            b += d;
            }
        bbb += z*b;
        v2 += z*b*b;
        }
```

```
    b = m*LOG( xx[0] ) + LOG( aa*( 1.0+xx[1]*bb ) );
    if ( b<=-150.0 )
        b = -150.0;
    det +=  b;
    v1 += bbb*bbb/( 1.0+xx[1]*bb );
    }
vo /= xx[0];
v1 *= xx[1];
v2 *= xx[2];
v2 /= xx[0];
return( 0.5*( det + vo - v1 - v2 ) );
}// End of MY_TYPE UTW_MLE::func(...)


void UTW_MLE::DoANOVA( void )
{
UTW_ANOVA::DoIt();
}// End of UTW_MLE::DoANOVA()


void UTW_MLE::GetPlotData( float *g, short n, float *xx,
                  float lx, float ux, short Obs )
{
float  dx, c;
dx = ( ux-lx )/Obs;
xx[ n ] = lx;
g[Obs] = -1.0E+6;
c = 1.0E+10;
for ( short i=0;i<Obs;i++ ) {
    g[i] = func( xx );
    if ( g[i]<c ) {
        g[Obs] = xx[n];
        c = g[i];
        }
    xx[n] += dx;
    }
}// End of void UTW_MLE::GetPlotData(...)


UTW_MM::UTW_MM( char *DataFile, USHORT _flag ):
UTW_MLE( DataFile, _flag )
{
short i, j, sum;
flag |= MLEMode;

missing = new short[ NoOfDam ];
Obsptr = new short[ NoOfDam ];
Sum_Yij = new float[ NoOfDam ];
if ( !missing || !Obsptr || !Sum_Yij ) {
    TW_Error = BadMemory;
    return;
    }

for ( i=0, sum=0;i<NoOfDam;i++ )
    {
    Sum_Yij[ i ] = 0.0;
    for ( j = 0; j< DamToOffspring[ i ]; j ++ )
        Sum_Yij[ i ] += Obs[ sum + j ];
```

```
    Obsptr[ i ] = sum;
    sum += DamToOffspring[ i ];

    if ( feof( in ) ) {
        TW_Error = BadFile;
        return;
        }

    fscanf( in, "%d", &j );
    if ( j<0 || j>1000 )
        {
        TW_Error = BadFile;
        return;
        }
    missing[i] = j;
    }

fscanf( in, "%d", &FullNoOfSir );
if ( FullNoOfSir<0 || FullNoOfSir>1000 )
    {
    TW_Error = BadFile;
    return;
    }
fscanf( in, "%d", &FullNoOfDam );
if ( FullNoOfDam<0 || FullNoOfDam>1000 )
    {
    TW_Error = BadFile;
    return;
    }
fscanf( in, "%d", &FullNoOfSib );
if ( FullNoOfSib<0 || FullNoOfSib>1000 )
    {
    TW_Error = BadFile;
    return;
    }

for (i = 0; i < 4; i++)
    {
    fscanf( in, "%f", &TruncatedValue );
    target_parameters[i] = TruncatedValue;
    }

fscanf( in, "%f", &TruncatedValue );
SetNodes();
flag &= ~MLEMode;
}// End of UTW_MM::UTW_MM(...)


UTW_MM::~UTW_MM( void )
{
delete [] Sum_Yij;
delete [] Obsptr;
delete [] missing;
}// End of UTW_MM::~UTW_MM()


void UTW_MM::SetNodes( void )
{
MY_TYPE gx[10], gw[10];
NoOfNode = 10;
```

```
gx[0] = 0.98695326;  gw[0] = 0.033335672;
gx[1] = 0.93253168;  gw[1] = 0.07472567;
gx[2] = 0.83970478;  gw[2] = 0.10954318;
gx[3] = 0.71669770;  gw[3] = 0.13463336;
gx[4] = 0.57443717;  gw[4] = 0.14776211;
gx[5] = 0.42556283;  gw[5] = 0.14776211;
gx[6] = 0.28330230;  gw[6] = 0.13463336;
gx[7] = 0.16029522;  gw[7] = 0.10954318;
gx[8] = 0.06746832;  gw[8] = 0.07472567;
gx[9] = 0.01304674;  gw[9] = 0.033335672;

   for ( short i=0;i<NoOfNode;i++ ) {
       dt[i] = ( 1.0- gx[i] )/gx[i];
       Node[i] = LOG( gw[i] ) - 0.5*dt[i]*dt[i] - 2.0*LOG( gx[i] );
       }
   }// End of void UTW_MM::SetNodes()


   MY_TYPE UTW_MM::func( float *xx )
   {
   if ( flag&MLEMode )
       return UTW_MLE::func( xx );

   short  ithSir, Dam_i, i;
   MY_TYPE f, ff;

   for ( i=0;i<3;i++ )                  // to easy computation below
       xx[i] = sqrt(xx[i]);

   f = Dam_i = 0;
   for ( ithSir=0; ithSir<NoOfSir; ithSir++ )
       {
       ff = IntgrV( xx, ithSir, Dam_i );
       if ( ff > EP )
           f -= LOG( ff );
       else
           f -= TINY;
       Dam_i += SirToDam[ithSir];
       }

   if ( flag&MissSirIncl && FullNoOfSir > NoOfSir )
       {
       ff = (MY_TYPE)(FullNoOfSir - NoOfSir) * IntgrV( xx );
       if ( ff > EP )
           f -= LOG( ff );
       else
           f -= TINY;
       }

   for ( i=0;i<3;i++ )                  // recovered to sigma^2
       xx[i] *= xx[i];

   return f;
   }// End of MY_TYPE UTW_MM::func(...)


   MY_TYPE UTW_MM::IntgrV( float *xx, short ithSir, short Dam_i )
```

```
{
short j, i;
MY_TYPE ss, c0, c1, s;

ss = 0.0;
for ( i=0;i<NoOfNode;i++ )
    {
    c0 = c1 = Node[i];
    for ( j=0;j<SirToDam[ithSir];j++ )
        {
        s = IntgrT( dt[i], xx, missing[Dam_i+j], Dam_i+j );
        if ( s > EP )
            c0 += LOG( s );
        else
            c0 += TINY;

        s = IntgrT( -dt[i], xx, missing[Dam_i+j], Dam_i+j );
        if ( s > EP )
            c1 += LOG( s );
        else
            c1 += TINY;
        }

    if ( flag&MissDamIncl && FullNoOfDam > j)
        {
        s = IntgrT( dt[i], xx );
        if ( s > EP )
            c0 += (MY_TYPE) (FullNoOfDam - j) * LOG( s );
        else
            c0 += (MY_TYPE) (FullNoOfDam - j) * TINY;

        s = IntgrT( -dt[i], xx );
        if ( s > EP )
            c1 += (MY_TYPE) (FullNoOfDam - j) * LOG( s );
        else
            c1 += (MY_TYPE) (FullNoOfDam - j) * TINY;
        }

    if ( c0 > TINY )
        ss += EXP( c0 );
    else
        ss += EP;

    if ( c1 > TINY )
        ss += EXP( c1 );
    else
        ss += EP;

    }

return  0.5*ss;
}// End of MY_TYPE UTW_MM::IntgrV(...)


MY_TYPE UTW_MM::IntgrV( float *xx )
{
short  j, i;
MY_TYPE ss, s, c;

ss = 0.0;
```

```
for ( i=0; i<NoOfNode; i++ )
    {
    c = Node[ i ];
    s = IntgrT( dt[i], xx );
    if ( s > EP )
        c += (MY_TYPE) (FullNoOfDam) * LOG( s );
    else
        c += (MY_TYPE) (FullNoOfDam) * TINY;
    ss += EXP( c );

    c = Node[ i ];
    s = IntgrT( -dt[i], xx );
    if ( s > EP )
        c += (MY_TYPE) (FullNoOfDam) * LOG( s );
    else
        c += (MY_TYPE) (FullNoOfDam) * TINY;
    ss += EXP( c );
    }

return  0.5*ss;
}// End of MY_TYPE UTW_MM::IntgrV(...)


MY_TYPE UTW_MM::IntgrT( MY_TYPE dv, float *xx )
{
short node_i;
MY_TYPE p, d, s;

s = 0.0;
for ( node_i=0; node_i < NoOfNode; node_i++ )
    {
    p = Phi( ( TruncatedValue - xx[3] - xx[1]*dv -
xx[2]*dt[node_i] ) / xx[0] );
    if ( p > 0.0 )
        {
        d = Node[node_i] + FullNoOfSib * LOG( p );
        if ( d > TINY )
            s += EXP( d );
        else
            s += EP;
        }

    p = Phi( ( TruncatedValue - xx[3] - xx[1]*dv +
 xx[2]*dt[node_i] ) / xx[0] );
    if ( p > 0.0 )
        {
        d = Node[node_i] + FullNoOfSib * LOG( p );
        if ( d > TINY )
            s += EXP( d );
        else
            s += EP;
        }
    }

return 0.5*s;
}// End of MY_TYPE UTW_MM::IntgrT(...)


MY_TYPE UTW_MM::IntgrT( MY_TYPE dv, float *xx, short miss,
```

```
  short Dam_ij )
{
short  k, node_i;
short  obs_by_Dam_ij = DamToOffspring[ Dam_ij ];
float * y_ij = &Obs[ Obsptr[ Dam_ij ] ];
MY_TYPE w, z, invxx0, c, d, e, s;

invxx0 = 1.0/xx[0];
e = obs_by_Dam_ij;
e *= LOG( xx[0] );

s = 0.0;
for ( node_i=0; node_i<NoOfNode; node_i++ )
    {
    w = ( xx[3] + xx[1]*dv + xx[2]*dt[node_i] )*invxx0;
    c = Phi( TruncatedValue*invxx0 - w );
    if ( c > 0.0 )
        {
        z = 0.0;
        for ( k=0; k<obs_by_Dam_ij; k++ )
            {
            d = y_ij[ k ] * invxx0 - w;
            z += (d * d);
            }
        d = Node[node_i] + miss * LOG( c ) - 0.5 * z - e;
        if ( d>TINY )
            s += EXP( d );
        else
            s += EP;
        }

    w = ( xx[3] + xx[1]*dv - xx[2]*dt[node_i] )*invxx0;
    c = Phi( TruncatedValue*invxx0 - w );
    if ( c > 0.0 )
        {
        z = 0.0;
        for ( k=0; k<obs_by_Dam_ij; k++ )
            {
            d = y_ij[ k ] * invxx0 - w;
            z += (d * d);
            }
        d = Node[node_i] + miss * LOG( c ) - 0.5 * z - e;
        if ( d>TINY )
            s += EXP( d );
        else
            s += EP;
        }
    }
return 0.5*s;
}// End of MY_TYPE UTW_MM::IntgrT(...)
```

This source file is for users interface only and not necessary for computation.

```
#include "\emx\ui\ui.hpp"
#include "mutw.hpp"
#include <conio.h>
#include <io.h>
#include <math.h>
#include <stdio.h>
```

```
#include <stdlib.h>
#include <string.h>

TEXT  *msg;
char * names[4] =
    {
     {"sigmao^2"},
     {"sigma1^2"},
     {"sigma2^2"},
     {"mean"}
    };


class MUTW_ICON : public WINMANAGER
{
public:
    MUTW_ICON( char *_FileName );
    MUTW_ICON( float target_parameters[], float _TrimRate,
int _Randseed );
    ~MUTW_ICON( void );
    short Event( const EVENT& event );
    void  Show( const Boolean DrawIt=TRUE );
private:
    void MUTW_SETUP( void );
    WRAP_BUTTON *mode;
    EMX_WINDOW *plotwin;
    COMBOX *plot;
    COMBOX *options;
    NUMBER *sigmal[4], *sigmau[4], *vsigma[4], *intv;
    EMX_WINDOW *genwin;
    BUTTON *gen;
    NUMBER *Sigma[4], *Ratio, *RandSeed;
    STRING *savefile;
    UTW_MM  *mlem;
    TWG     *twg;
    char    buf[256];
    float   lx[5], ux[5], xx[5], tmpx[5], vRatio, vIntv, *g;
    short   plotpts, rSeed;
    Boolean gen_locked;
};


MUTW_ICON::MUTW_ICON( char * _FileName )
    : WINMANAGER( 6, " Unbalanced Two Way Nested Classification
(Maximum Likelihood) " )
{
if ( !_FileName || (_FileName && *_FileName &&
access( _FileName, 0)) )
    Fatal_Error( "Bad File Name" );

gen_locked = TRUE;
mlem = new UTW_MM( _FileName );

if ( TW_Error>0 )
    {
    if ( TW_Error&BadFile )
        Fatal_Error( "DataFile is Bad" );
    else
        Fatal_Error( "No Memory" );
```

```
        }
    MUTW_SETUP();
    }


    MUTW_ICON::MUTW_ICON( float _xx[], float _TrimRate,
  int _RandSeed )
        : WINMANAGER( 6, " Unbalanced Two Way Nested
Classification (Maximum Likelihood) " )
    {
    gen_locked = FALSE;

    vRatio = _TrimRate;
    rSeed = _RandSeed;
    twg = new TWG( _xx[0], _xx[1], _xx[2], _xx[3], vRatio, rSeed );
    if ( twg->DoIt() )
        mlem = new UTW_MM( "tw.mis" );
    else
        Fatal_Error("Trim Too Heavy");

    MUTW_SETUP();
    }


    void MUTW_ICON::MUTW_SETUP( void )
    {
    short i, j;
    float c;
    const float * x = mlem->GetTargetParameters();
    msg = new TEXT( "",  5, 65, 17, 10, 256, " Message Board " );

    vIntv = 0.375;
    for ( i=0;i<4;i++ )
        {
        c = x[i]*vIntv;
        if ( c<0.0 )
            c = -c;
        xx[i] = x[i];
        lx[i] = x[i] - c;
        ux[i] = x[i] + c;
        if ( i<3 && lx[i]<=0.005 )
            lx[i] = 0.005;
        if ( i<3 && ux[i]<=0.01 )
            ux[i] = lx[i] + 0.01;
        }
    if ( fabs( x[3] )<0.05 )
        {
        lx[3] = -5.0;
        ux[3] =  5.0;
        }

    strcpy( buf, "  MIS  " );
    i = 1;
    strcpy( &buf[ 8 * i ++ ], "  MLE  " );
    strcpy( &buf[ 8 * i ++ ], " ANOVA " );
    mode = new WRAP_BUTTON( i, buf, 1, 8, 21 );

    for ( i=0, j=0; i < 4; i++, j += 8)
```

```
    sprintf( &buf[j], " %s  ", names[i] );

plotwin = new EMX_WINDOW( 20, 10, 50, 6, 4, " Op~timization " );

*plotwin
    + ( plot = new COMBOX( "   P~lot  ", buf, 6, 8, 10, 0 ) )
    + ( sigmal[0] = new NUMBER( lx[0], "%.2f", 1, 14, 10,  7, 7,
" low  " ) )
    + ( sigmau[0] = new NUMBER( ux[0], "%.2f", 1, 14, 10, 22, 7,
  " high " ) )
    + ( vsigma[0] = new NUMBER( xx[0], "%.2f", 1, 14, 10, 37, 7,
buf ) )
    + ( sigmal[1] = new NUMBER( lx[1], "%.2f", 1, 14, 11,  7, 7,
" low  " ) )
    + ( sigmau[1] = new NUMBER( ux[1], "%.2f", 1, 14, 11, 22, 7,
  " high " ) )
    + ( vsigma[1] = new NUMBER( xx[1], "%.2f", 1, 14, 11, 37, 7,
  &buf[8] ) )
    + ( sigmal[2] = new NUMBER( lx[2], "%.2f", 1, 14, 12,  7, 7,
  " low  " ) )
    + ( sigmau[2] = new NUMBER( ux[2], "%.2f", 1, 14, 12, 22, 7,
   " high " ) )
    + ( vsigma[2] = new NUMBER( xx[2], "%.2f", 1, 14, 12, 37, 7,
   &buf[16] ) )
    + ( sigmal[3] = new NUMBER( lx[3], "%.2f", 1, 14, 13,  7, 7,
    " low  " ) )
    + ( sigmau[3] = new NUMBER( ux[3], "%.2f", 1, 14, 13, 22, 7,
    " high " ) )
    + ( vsigma[3] = new NUMBER( xx[3], "%.2f", 1, 14, 13, 37, 7,
    &buf[24] ) );


strcpy( buf, " MissSir " );
strcpy( &buf[ 10 ], " MissDam " );

*plotwin
    + ( options = new COMBOX( "  Op~tion ", buf, 4, 8, 29, 0 ) )
    + ( intv = new NUMBER( vIntv, "%.3f", 1, 9, 8, 40, 5, " ~r" ) )
    + mode;

buf[0] = buf[1] = 1;
options->SetItemStatus( buf );
*this + plotwin;

for ( i=0, j=0; i < 4; i++, j += 8)
    sprintf( &buf[j], " %s  ", names[i] );

if ( !gen_locked ) {
    genwin = new EMX_WINDOW(  9, 13, 29, 2, 42,
  " ~Data Generation " );
    *genwin
        + ( gen = new BUTTON( "   Re~generate Data  ", 1, 4, 46 ) )
        + ( Sigma[0] = new NUMBER( xx[0], "%.2f", 1, 16,  6, 49,
        7, buf ) )
        + ( Sigma[1] = new NUMBER( xx[1], "%.2f", 1, 16,  7, 49,
  7, &buf[8] ) )
```

```
            + ( Sigma[2] = new NUMBER( xx[2], "%.2f", 1, 16,  8, 49,
      7, &buf[16] ) )
            + ( Sigma[3] = new NUMBER( xx[3], "%.2f", 1, 16,  9, 49,
      7, &buf[24] ) )
            + ( RandSeed = new NUMBER( rSeed, 1, 16, 10, 49, 5,
" R~andSeed " ) )
            + ( Ratio    = new NUMBER( vRatio,"%.2f", 1, 16, 11, 49,
    4, " M~isRatio " ) )
            + ( savefile = new STRING( "a001",         1, 16, 12, 49,
    8, " Sa~ve as" ) );
        *this + genwin;
        }

    SetUp();
    wrefresh( CmdWin );
    plotpts = 18;
    g = new float[ plotpts+2 ];
    }// End of MUTW_ICON::MUTW_ICON()


    MUTW_ICON::~MUTW_ICON( void )
    {
    delete mlem;
    if ( !gen_locked )
        delete twg;
    delete [] g;
    delete msg;
    }// End of MUTW_ICON::~MUTW_ICON()


    void MUTW_ICON::Show( const Boolean DrawIt )
    {
    WINMANAGER::Show( DrawIt );
    msg->Show( DrawIt );
    wrefresh( CmdWin );
    }// End of void MUTW_ICON::Show(...);


    short MUTW_ICON::Event( const EVENT& event )
    {
    switch( event.rawKey ) {
        case Key_DOWN:
        if ( CurTask()==0 &&
             ( plotwin->CurTask()>=1 &&
               plotwin->CurTask()<=9 ) ) {
            WINMANAGER::Event( event );
            WINMANAGER::Event( event );
            }
        break;

        case Key_UP:
        if ( CurTask()==0 &&
             ( plotwin->CurTask()>=4 &&
               plotwin->CurTask()<=12 ) ) {
            WINMANAGER::Event( event );
            WINMANAGER::Event( event );
            }
        break;
```

```
        default:
        break;
        }

    EVENT tevent;
    short ccode = WINMANAGER::Event( event );
    float c;

    short i, curTask = CurTask();
    switch ( ccode ) {
        case Key_SELECTED:
        if ( curTask==0 )
            {
            for ( i=0;i<4;i++ )
                {
                lx[i] = atof(sigmal[i]->GetData());
                ux[i] = atof(sigmau[i]->GetData());
                tmpx[i] = atof( vsigma[i]->GetData());
                if ( i<3 && lx[i]<=0.005 )
                   lx[i] = 0.005;
                if ( i<3 && ux[i]<=0.01 )
                   ux[i] = lx[i] + 0.01;
                }
            if ( fabs( tmpx[3] )<0.05 ) {
                lx[3] = -5.0;
                ux[3] =  5.0;
                }

            if ( mode->GetItem()==0 )
                {
                char * ptr = options->GetItemStatus();

                i = 0;
                if ( ptr[0] )
                    i |= MissSirIncl;
                if ( ptr[1] )
                    i |= MissDamIncl;

                mlem->SetFlag( i );
                }

            if ( mode->GetItem()==1 )
                mlem->SetFlag( MLEMode );

                                         // plot graphics
            if ( plotwin->CurTask()==0 ) {
                sprintf( buf, "Searching for %s ....",
names[ plot->GetItem()->code ] );
                msg->SetData( buf );
                msg->Show();
                i = plot->GetItem()->code;
                mlem->GetPlotData( g, i, tmpx, lx[i], ux[i], plotpts );
                if ( PlotResults( g, plotpts, lx[i], ux[i] ) ) {
                    if ( g[plotpts]!=-1.0E+6 ) {
                        sprintf( buf, "%.2f", g[plotpts] );
                        vsigma[i]->SetData( buf );
                        vIntv = atof( intv->GetData() );
                        c = g[ plotpts ]*vIntv;
```

```
                    if ( c<0.0 ) c = -c;
                    lx[i] = g[ plotpts ] - c;
                    sprintf( buf, "%.2f", lx[i] );
                    sigmal[i]->SetData( buf );
                    ux[i] = g[ plotpts ] + c;;
                    sprintf( buf, "%.2f", ux[i] );
                    sigmau[i]->SetData( buf );
                    if ( i==3 && fabs( g[plotpts] )<0.05 ) {
                        ux[i] =  1.0;
                        lx[i] = -1.0;
                        }
                    }
                msg->SetData("");
                }
            else
                msg->SetData("Object function values cann't be
differentialized!" );
            }

        Show();
        }
                    // generate a new data set
    if ( !gen_locked && curTask==1 && genwin->CurTask()==0 )
        {

        delete mlem;
        delete twg;

        vIntv = atof( intv->GetData() );
        for ( i=0;i<4;i++ )
            {
            xx[i] = atof( Sigma[i]->GetData() );
            if ( i<3 && xx[i]<=0.005 )
               xx[i] = 0.005;
            c = xx[i]*vIntv;
            if ( c<0.0 )
                c = -c;
            lx[i] = xx[i] - c;
            ux[i] = xx[i] + c;

            if ( fabs( xx[3] ) < 0.05 )
                {
                lx[3] = -5.0;
                ux[3] =  5.0;
                }
            sprintf( buf,"%.2f", lx[i] );
            sigmal[i]->SetData( buf );
            sprintf( buf,"%.2f", ux[i] );
            sigmau[i]->SetData( buf );
            }
        Show();
        rSeed = atoi( RandSeed->GetData() );
        vRatio = atof( Ratio->GetData() );
        twg = new TWG( xx[0], xx[1], xx[2], xx[3], vRatio, rSeed );
        if ( twg->DoIt() )
            mlem = new UTW_MM( "tw.mis" );
        else
            Fatal_Error("Trim Too Heavy");
```

```
            tevent.rawKey = Key_WINMANAGER;
            WINMANAGER::Event( tevent );
            tevent.rawKey = Key_RETURN;
            WINMANAGER::Event( tevent );
            tevent.rawKey = 't';
            WINMANAGER::Event( tevent );
            }
        break;

        case Key_RETURN :
        if ( curTask==0 && mode->GetItem()==2 )
            {
            mlem->DoANOVA();
            sprintf( buf,"ANOVA: sigmao^2=%.2f  sigma1^2=%.2f
sigma2^2=%.2f  mu=%.2f\rHit <Enter> to continue...",
                     mlem->GetResults()[0], mlem->GetResults()[1],
                     mlem->GetResults()[2], mlem->GetResults()[3] );
            msg->SetData( buf );
            msg->Show();
            mode->SetItem( 1 );
            mode->Show();
            wgetch( CmdWin );
            msg->SetData( "" );
            msg->Show();
            wrefresh( CmdWin );
            }
                                              // reset trim rate
        if ( !gen_locked && curTask==1 && genwin->CurTask()==6 ) {
            if ( fabs( vRatio-atof( Ratio->GetData() ) )<0.001 )
                return ccode;
            else
                vRatio = atof( Ratio->GetData() );
            twg->SetTrimRate( vRatio );
            if ( twg->DoIt() ) {
                delete mlem;
                mlem = new UTW_MM( "tw.mis" );
                }
            else {
                msg->SetData("Error: Trim Too Heavy");
                msg->Show();
                flash();
                return ccode;
                }
            msg->SetData("Data's missing number has been changed.");
            msg->Show();
            tevent.rawKey = Key_WINMANAGER;
            WINMANAGER::Event( tevent );
            tevent.rawKey = Key_RETURN;
            WINMANAGER::Event( tevent );
            tevent.rawKey = 't';
            WINMANAGER::Event( tevent );
            }
                                                    // save data to files
        if ( !gen_locked && curTask==1 && genwin->CurTask()==7 ) {
            char buf[60];
            sprintf( buf, "%s.dat", savefile->GetData() );
            twg->Write( buf );
```

```
            sprintf( buf, "%s.mis", savefile->GetData() );
            twg->WriteMis( buf );
            sprintf( buf, "Data are saved into %s.dat & %s.mis!",
                     savefile->GetData(), savefile->GetData() );
            msg->SetData( buf );
            msg->Show();
            }

        break;

        default:
        break;
        }

return ccode;
}// End of short MUTW_ICON::Event()


main( int argc, char *argv[] )
{
int i;
MUTW_ICON  *icon = NULL;

// This part is necessary for any modules in EMX involved
   curses.lib!
initscr();
if ( start_color() != OK )
    fatal_error("Couldn't Start Color" );
cbreak();
noecho();
nonl();
SetUpColors();
CmdWin = newwin( 1, SCRCOL, 24, 0 );
keypad( CmdWin, TRUE );
wmove( CmdWin, 0, 0 );
//****************************** END *******************************

if (argc == 2)
    {
    if (strncmp( argv[1], "-f", 2 ) == 0)
        icon = new MUTW_ICON( &argv[1][2] );
    }
else if (argc > 5)
    {
    int _RandSeed = 0;
    float _TrimRate = 0.7;
    float target_parameters[4];
    Boolean good_argv = TRUE;

    for (i = 0; i < 4; i ++)
        target_parameters[i] = atof( argv[i + 1] );

    for (int i = 5;  i < argc;  i++)
       {
        if (0 == strncmp( argv[i], "-r", 2 ))
           {
            _RandSeed = atoi( &argv[i][2] );
            if (_RandSeed < 0 || 32766 < _RandSeed)
```

```
                    good_argv = FALSE;
                }
            else if (0 == strncmp( argv[i], "-t", 2 ))
                {
                _TrimRate = atof( &argv[i][2] );
                if (_TrimRate < 0.0 || 0.95 < _TrimRate)
                    good_argv = FALSE;
                }
            }

        if (good_argv)
            icon = new MUTW_ICON( target_parameters, _TrimRate,
    _RandSeed );
        }

    if (!icon)
        {
        printf( "\n\n%s -fxxxx OR\n", argv[0] );
        printf( "%s (sigmao^2 sigma1^2 sigma2^2 mean [-rt])\n",
argv[0] );
        printf( "    -f: datafile (e.g. -ftw.dat)\n" );
        printf( "    -r: randseed (e.g. -r1000, from 0 to 32766)\n" );
        printf( "    -t: TrimRate (e.g. -t0.0, from 0.0 to 0.95)\n" );
        printf( "Note: when with -f option, other arguments are
ignored\n" );
        printf( "      randseed = 0 forces to randomize the
generation\n" );
        exit( 1 );
        }

    short  ccode;
    EVENT  event;
    do  {
        event.rawKey = wgetch( CmdWin );
        ccode = icon->Event( event );
        } while( ccode!=Key_EXIT );

    delete icon;
    // Always delete any curses.lib related objects before
       endwin()!!!!!
    refresh();
    endwin();
    return 0;
    }// End of main()
```

Bibliography

[1] Afifi, A. A., and Elashoff, R. M. (1966). Missing observations in multivariate statistics I: Review of the literature. J. Amer. Stat. Assoc. 61, 595-601.

[2] Ahrens, H., Kleffe, J & Tenzler, R. (1981). Mean square error comparison for MINQUE, ANOVA and two alternative estimators under the unbalanced one way random model. Biometrical J. 23, 323-342.

[3] Anderson, T. W. (1957). Maximum likelihood estimates for the multivariate normal distribution when some observation are missing. J. Amer. Stat. Assoc. 52, 200-203.

[4] Anderson, T. W. (1984). An Introduction to Multivariate Analysis, 2nd edn. John Wiley & Sons, New York.

[5] BMDP (1988). BMDP Statistical Software, Inc., 1440 Sepulveda Blvd, Los Angeles, California.

[6] Borardman, T. J. (1974). Confidence intervals for variance components a comparative Monte Carlo study. Biometrics 30, 251-262.

[7] Brownlee, K. A. (1953). Industrial Experimentation. Chemical Publishing Co.

[8] Burdick, R. K. and Graybill, F. A. (1984). Confidence intervals on linear combinations of variance components in the unbalanced one-way classification. Technometrics 26, 131-136.

[9] Burdick, R. K. and Graybill, F. A. (1988). The present status of confidence interval estimation on variance components in balanced and unbalanced random models. Commun. Stat.: Theory & Methods (Special Issue on Analysis of the Unbalanced Mixed Model) 17, 1165-1195.

[10] Chen, T. and Fienberg, S. E. (1974). Two-dimensional contingency tables with both completely and partially classified data. Biometrics 30, 629-642.

[11] Corbeil, R. R. and Searle, S. R. (1976). A comparison of variance component estimators. Biometrics 32, 779-791.

[12] Davis, P. and Rabinowitz, P. (1984). Methods of Numerical integration. Academic Press, New York.

[13] Dempster, A., Laird, N. and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. J. of the Royal statistical society, B, 39, 1 38.

bibitemDic:1991 DiCiccio, T. J. and Field, C. A. (1991). An accurate method for approximate conditional and Bayesian inference about linear regression models from censored data. Biometrika, 78, 903-910.

[14] Fellner, W. H. (1986). Robust Estimation of Variance Components. Technometrics 28, 51-60.

[15] Fuchs, C. (1982). Maximum likelihood estimation and model selection in contingency tables with missing data. J. Amer. Stat. Assoc. 77, 270-278.

[16] Graybill, F. A. (1976). Theory and Application of the Linear Statistical Model. Duxbury, North Scituate, Massachusetts.

[17] Graybill, F. A. and Wang, C. M. (1980). Confidence intervals on non-negative linear combinations of variance. J. Amer. Stat. Assoc. 75, 869-873.

[18] Hampel, F. R., Ronchetti, E. M., Rousseeuw, P. J., and Stahel, W. A. (1986). Robust Statistics: The Approach Based on Influence Functions. Wiley, New York.

[19] Hartley, H. O. and Hocking, R. R. (1971). The analysis of incomplete data. Biometrics 27, 783-823.

[20] Hartley, H. O. and Rao, J. N. K. (1967). Maximum likelihood estimation for the mixed analysis of variance model. Biometrika 54, 93-108.

[21] Harville, D. A. (1976). Extension of the Gauss-Markov theorem to include the estimation of random effects. Annals of statistics 4, 384-395.

[22] Harville, D. A. (1977). Maximum Likelihood Approaches to Variance Component Estimation and to Related Problems. J. Amer. Stat. Assoc. 72, 320-340.

[23] Henderson, C. R. (1973). Maximum Likelihood Estimation of Variance Components. Unpublished manuscript, Department of Animal Science, Cornell University, Ithaca, New York.

[24] Hill, B.M. (1965). Inference about variance components in the one-way model. J. Amer. Stat. Assoc. 60, 806-825.

[25] Hill, B.M. (1967). Correlated errors in the random model. J. Amer. Stat. Assoc. 62, 1387-1400.

[26] Hocking, R. R. and Smith, W. B. (1968). Estimation of parameters in the multivariate normal distribution with missing observations. J. Amer. Stat. Assoc. 63, 159-173.

[27] Huber, P. J. (1981). Robust Statistics. Wiley, New York.

[28] Ibrahim, J. G. (1990). Incomplete data in generalized linear models. J. Amer. Stat. Assoc. 85, 765-769.

[29] Khuri, A. I. and Sahai, H. (1985). Variance components analysis: a selective literature survey. Int. Stat. Rev., 53, 279-300.

[30] Laird, N. (1982). Computation of variance components using the EM algorithm. J. Stat. Comput. Simul. 14, 295-303.

[31] Lindsay, B.G. (1983.) The geometry of mixing likelihoods: A general theory. Annals of Statistics 11, 86-94.

[32] La Motte, L. R. (1973). Quadratic Estimation of Variance Components. Biometrics 29, 311-330.

[33] Little, R. and Rubin, D. (1987). Statistical Analysis with missing data, John Wiley & Sons, New York.

[34] McCullagh, P. and Nelder, J. A. (1989). Generalized Linear Models 2nd edn, Chapman and Hall, London.

[35] Miller, J. J. (1973). Asymptotic properties and computation of maximum likelihood estimates in the mixed model of the analysis of variance. Technical Report No. 12, Department of Statistics, Stanford University, Stanford, California.

[36] Miller, R. (1980). Survival Analysis. Wiley, New York.

[37] Patterson, H. D., and Thompson, R. (1971). Recovery of Interblock Information When Block Sizes are Unequal. Biometrika 58, 545-554.

[38] Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T. (1986). Numerical Recipes. Cambridge: Cambridge University Press.

[39] Pukelsheim, F. (1981). On the Existence of Unbiased Non-negative Estimates of Variance Covariance Components. Ann. Stat. 9, 293-299.

[40] Rao, C. R. (1970). Estimation of Heteroscedastic Variances in Linear Models. J. Amer. Stat. Assoc. 65, 161-172.

[41] Rao, C. R. (1971a). Estimation of Variance and Covariance Components. J. of Multivariate Analysis 1, 257-275.

[42] Rao, C. R. (1971b). Minimum Variance Quadratic Unbiased Estimation of Variance Components. J. of Multivariate Analysis 1, 445-456.

[43] Rao, C. R. (1972). Estimation of variance and covariance components in linear models. J. Amer. Stat. Assoc. 67, 112-115.

[44] Rao, C. R. (1973). Linear statistical inference and its applications. Wiley. New York.

[45] Rao, C. R. (1979). MINQUE theory and its relation to ML and MML estimation of variance components. Sankhya B 41, 138-153.

[46] Rao, C. R. and Kleffe, J. (1980). Estimation of Variance Components. Krishnaiah, P. R. Handbook of Statistics. North-Holland, Amsterdam, 1-40.

[47] Rao, C. R. and Kleffe, J. (1988). Estimation of Variance Components and Applications. North-Holland, Amsterdam.

[48] Robinson, G. K. (1991). That BLUP is a good thing- the estimation of random effects. Stat. Sci. 6, 15-51.

[49] Rubin, D. B. (1976). Inference and missing data. Biometrika 63, 581-592.

[50] Schafer, D. W. (1987). Covariate measurement error in generalized linear models. Biometrika 74, 385-391.

[51] Searle, S. R. (1970). Large sample variances of maximum likelihood estimators of variance components using unbalanced data. Biometrics 26, 505-524.

[52] Searle, S. R. (1971). Topics in variance components estimation. Biometrics 27, 1-76.

[53] Searle, S. R. (1982). Matrix Algebra Useful for Statistics. Wiley, New York.

[54] Searle, S. R. (1989). Variance components-some history and a summary account of estimation methods. J. Animal Breeding & Genetics 106, 1-29.

[55] Searle, S. R., Casella, G. and McCulloch, C. E. (1992). Variance components. John Wiley & Sons, New York.

[56] Swallow, W. H. and Monahan, J. F. (1984). Monte-Carlo comparison of ANOVA, MIVQUE,REML, and ML estimators of variance components. Technometrics 26, 47-57.

[57] Tanner, M. A. (1991). Tools for statistical inference: observed data and augmentation methods. Springer-Verlag, New York.

[58] Townsend, E. C. and Searle, S. R. (1971). Best quadratic unbiased estimation of variance components from unbalanced data in the one way classification. Biometrics 27, 643-657.

[59] Westfall, P. (1987). A Comparison of Variance Component Estimates for Arbitrary Underlying Distributions. J. Amer. Stat. Assoc. 82, 866-874.

[60] Yates, F. (1940). The Recovery of Inter-Block Information in Balanced Incomplete Block Designs. Ann. Eugenics 10, 317-325.

[61] Yates, F., and Zacopancy, I. (1935). The Estimation of the Efficiency of Sampling with Special Reference to Sampling for Yield in Cereal Experiments. J. Agric. Sci. 25, 545-577.

[62] Yu, H., Searle, S. R. and McCulloch, C. E. (1991). Properties of maximum likelihood estimators of variance components in the one-way classification model, balanced data. Technical Report BU-1134-M, Biometrics Unit, Cornell University, Ithaca, New York.

[63] Zielinski, W. (1986). On Robust Estimation of Variance Components. Probab. Stat. 7, 95-102.